Crystals that think
about how they're growing

David Doty

joint work with Damien Woods, Erik Winfree, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin

Computing with Unconventional Technologies Workshop, October 18, 2021

Caltech Inria Paris UC Davis Harvard

Acknowledgements

Caltech

Inria Paris

UC Davis

Harvard

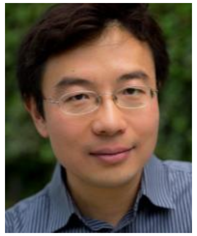
Damien Woods
(co-first author)

Erik Winfree
(PI)

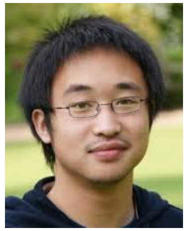
co-authors

Cameron Myhrvold

Peng Yin



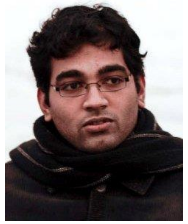
Felix Zhou



Joy Hui

lab/science help

Sungwook Woo
Sarina Mohanty
Deborah Fygenson
Mingjie Dai
Chris Thachuk
Jongmin Kim
Bryan Wei
Ashwin Gopinath



Constantine Evans
Niranjan Srinivas
Yannick Rondolez
Nikhil Gopalkrishnan
Nadine Dabby
Paul Rothemund
Cody Geary

side project

Chris Thachuk
Namita Sarraf
Anya Mitskovets
Damien Woods
Pierre-Etienne Meunier
Constantine Evans

Diverse and robust molecular algorithms using reprogrammable DNA self-assembly.
Damien Woods†, David Doty†, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, Erik Winfree.
Nature 2019. †*These authors contributed equally*.

Hierarchy of abstractions

➡ **Bits:**	**Boolean circuits compute**
Tiles:	Tile self-assembly implements circuits
DNA:	DNA strands implement tiles

Harmonious arrangement

0

1

1

0

1

1

Harmonious arrangement

0

1

1

0

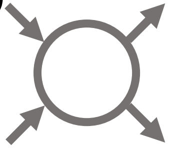
1

1

Harmonious arrangement

0	1
1	1
1	1
0	1
1	0
1	0

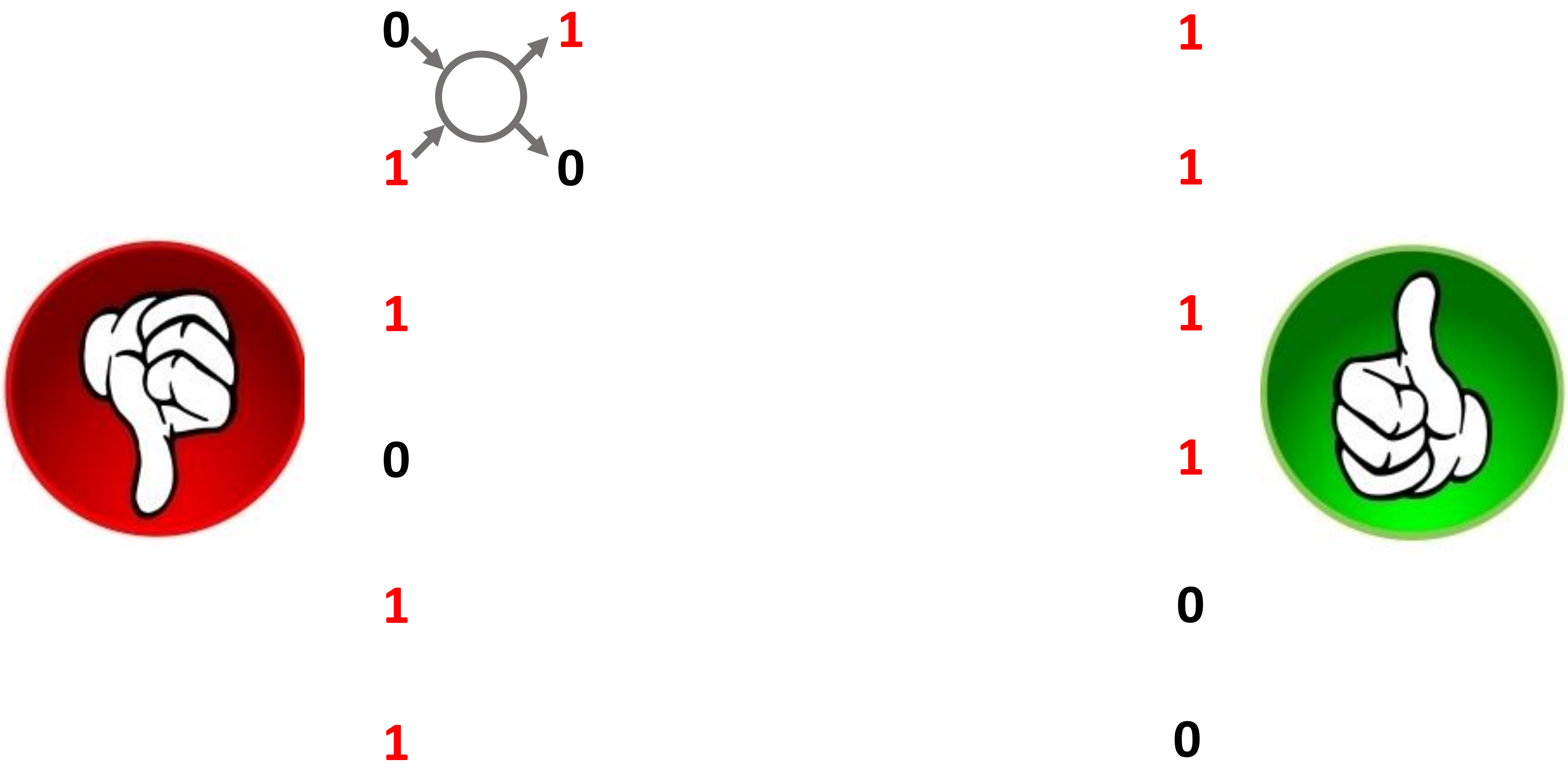
Harmonious arrangement



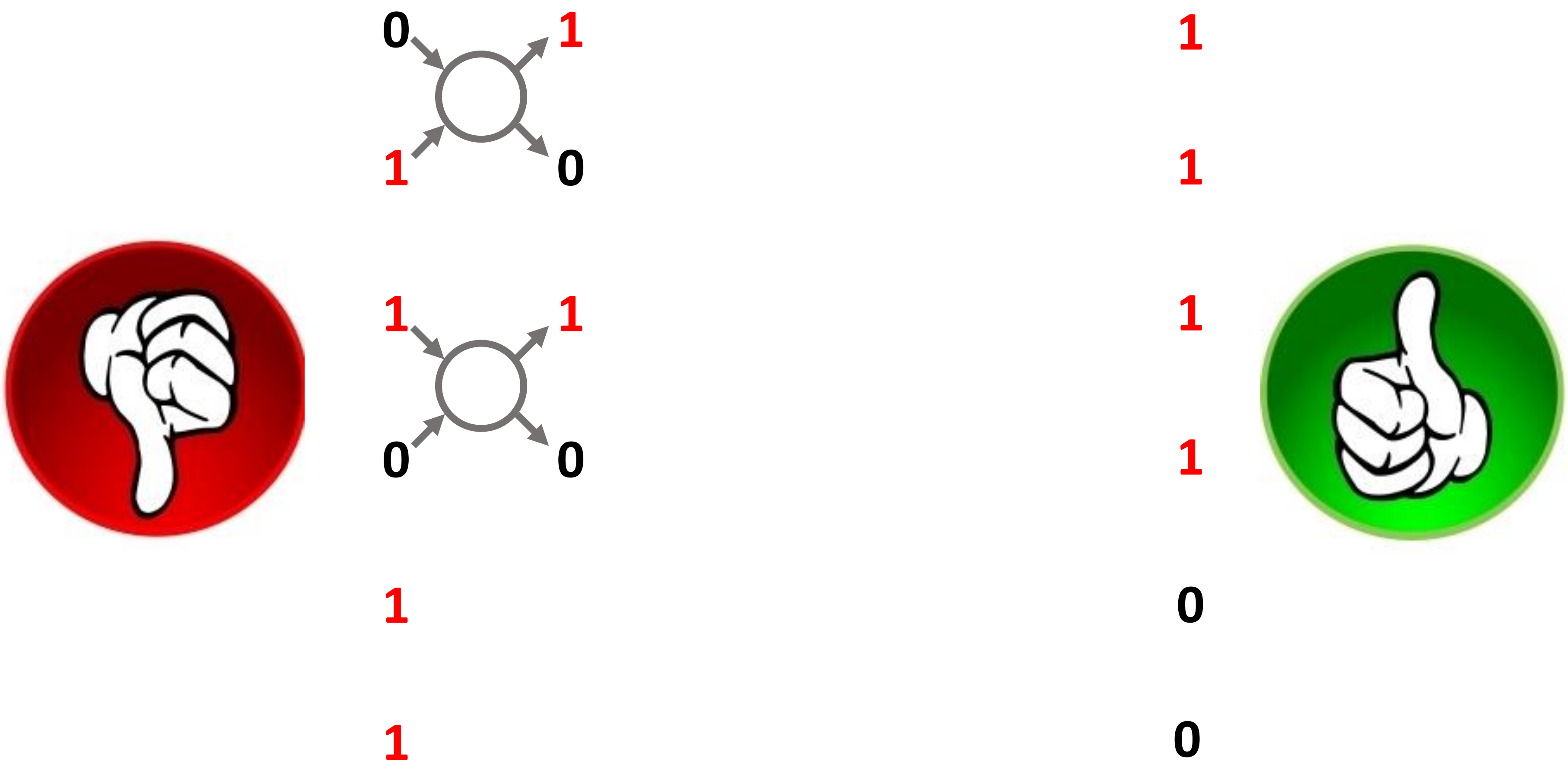
0
1
1
0
1
1

1
1
1
1
0
0

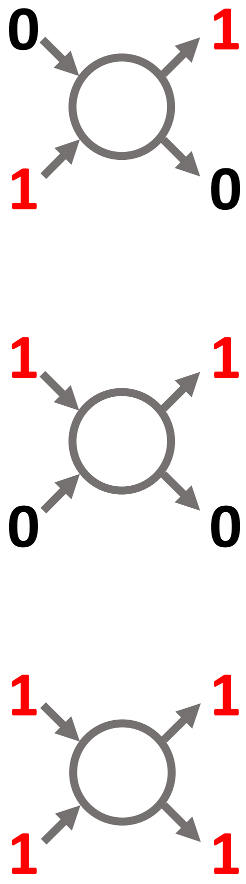
Harmonious arrangement



Harmonious arrangement



Harmonious arrangement



1

1

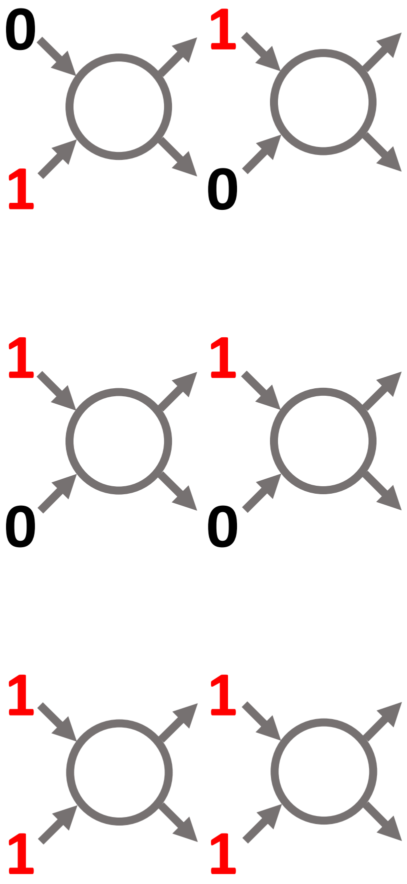
1

1

0

0

Harmonious arrangement



1

1

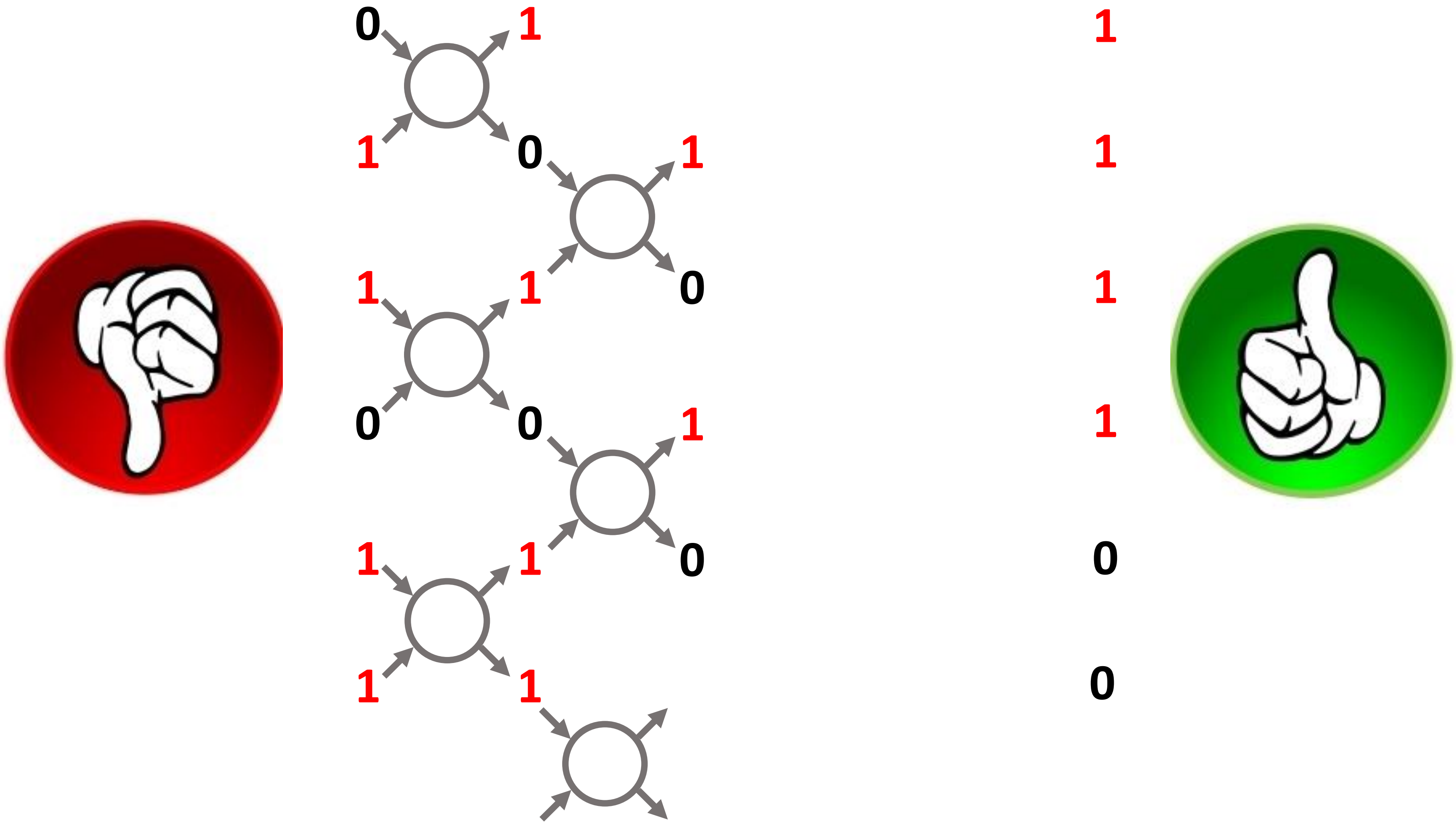
1

1

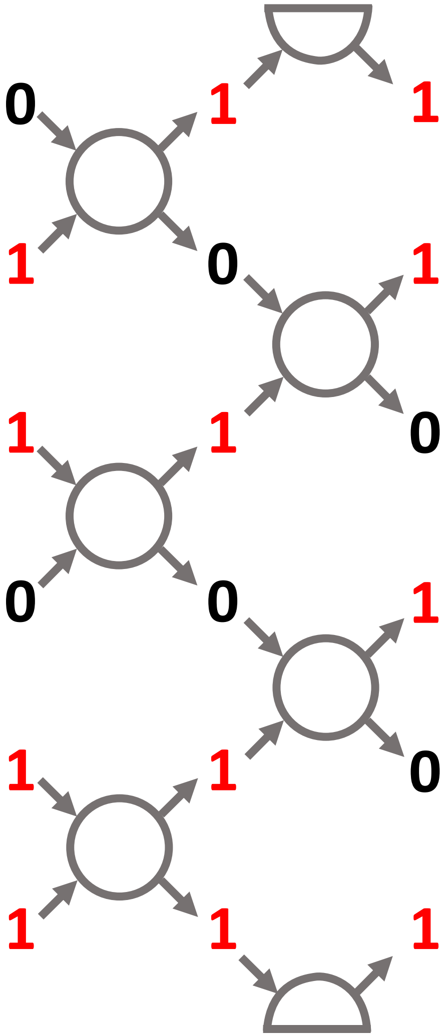
0

0

Harmonious arrangement

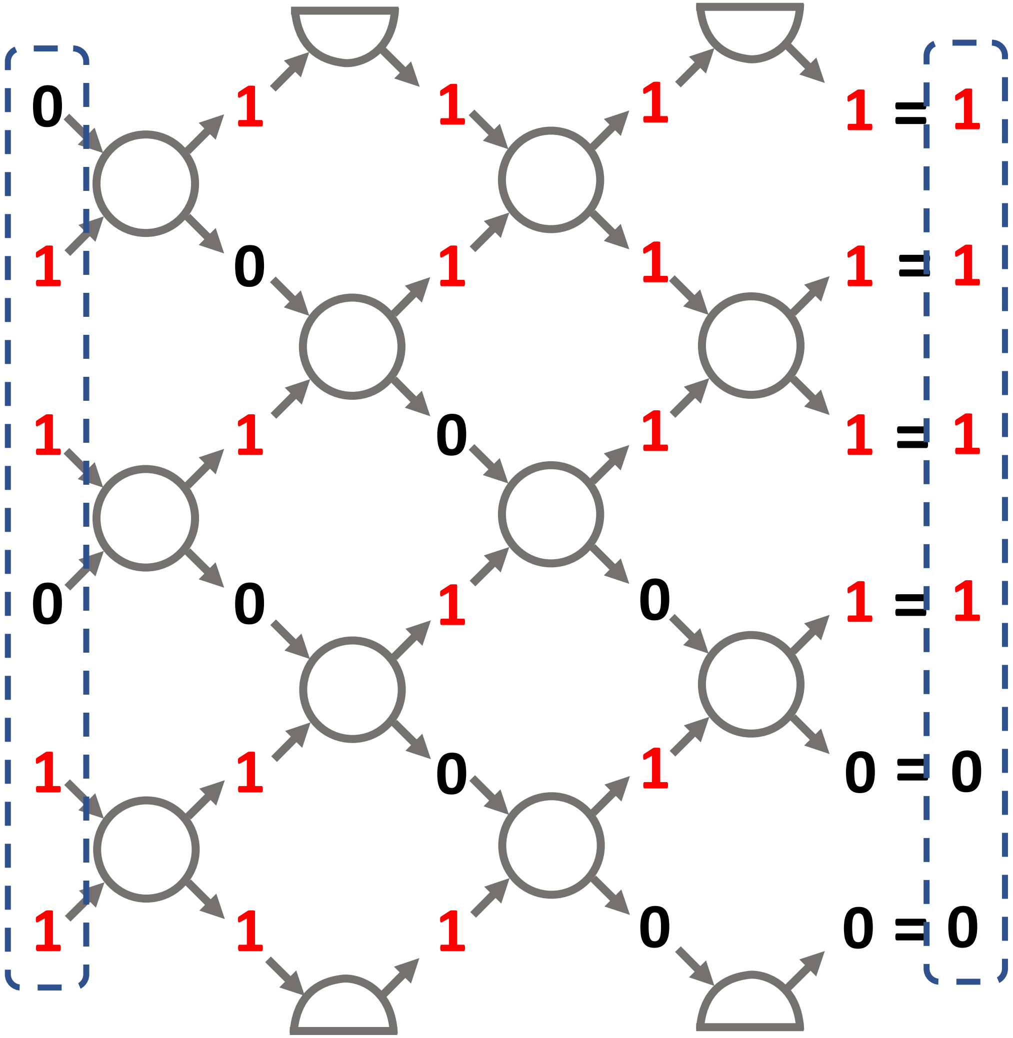


Harmonious arrangement



1
1
1
1
0
0

Harmonious arrangement



a.k.a. sorting

Odd bits

1

0

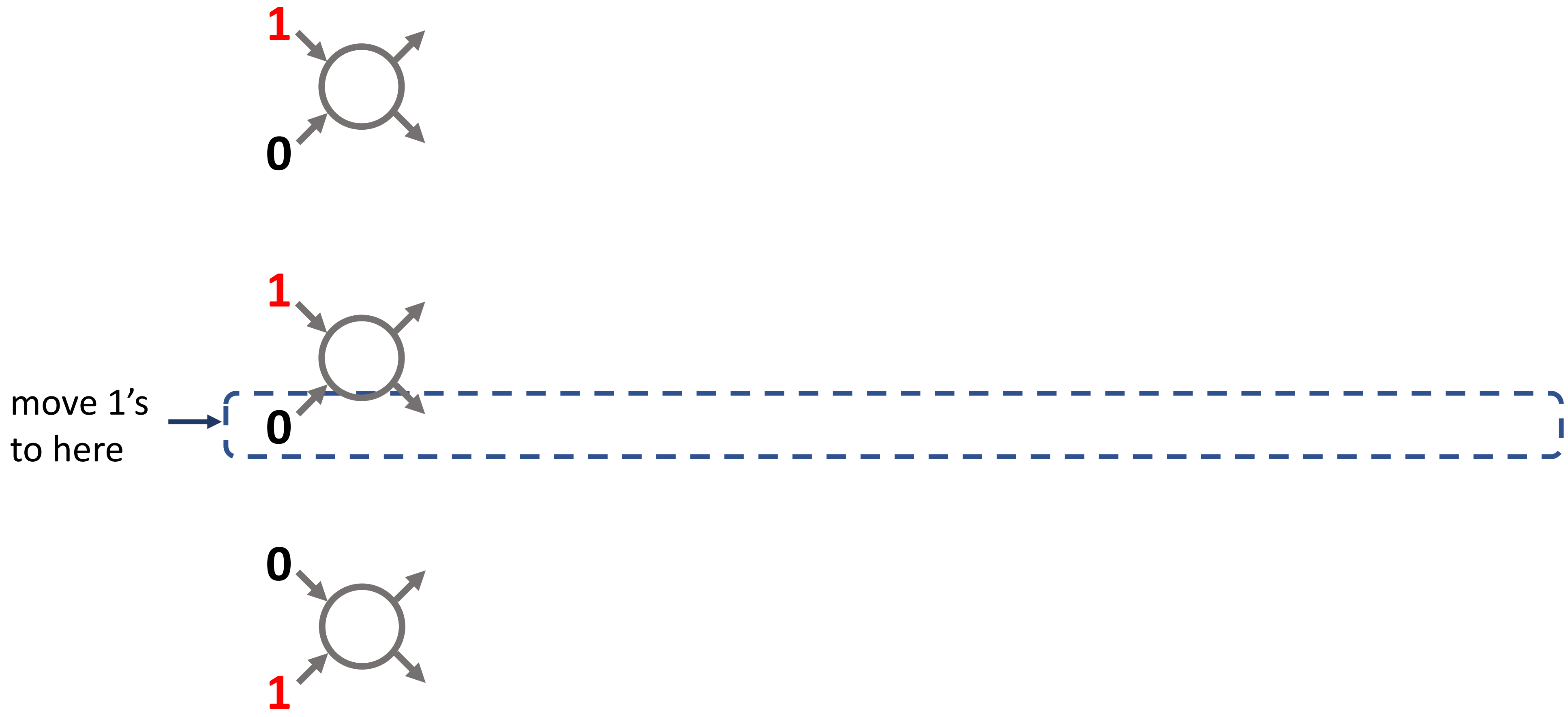
1

0

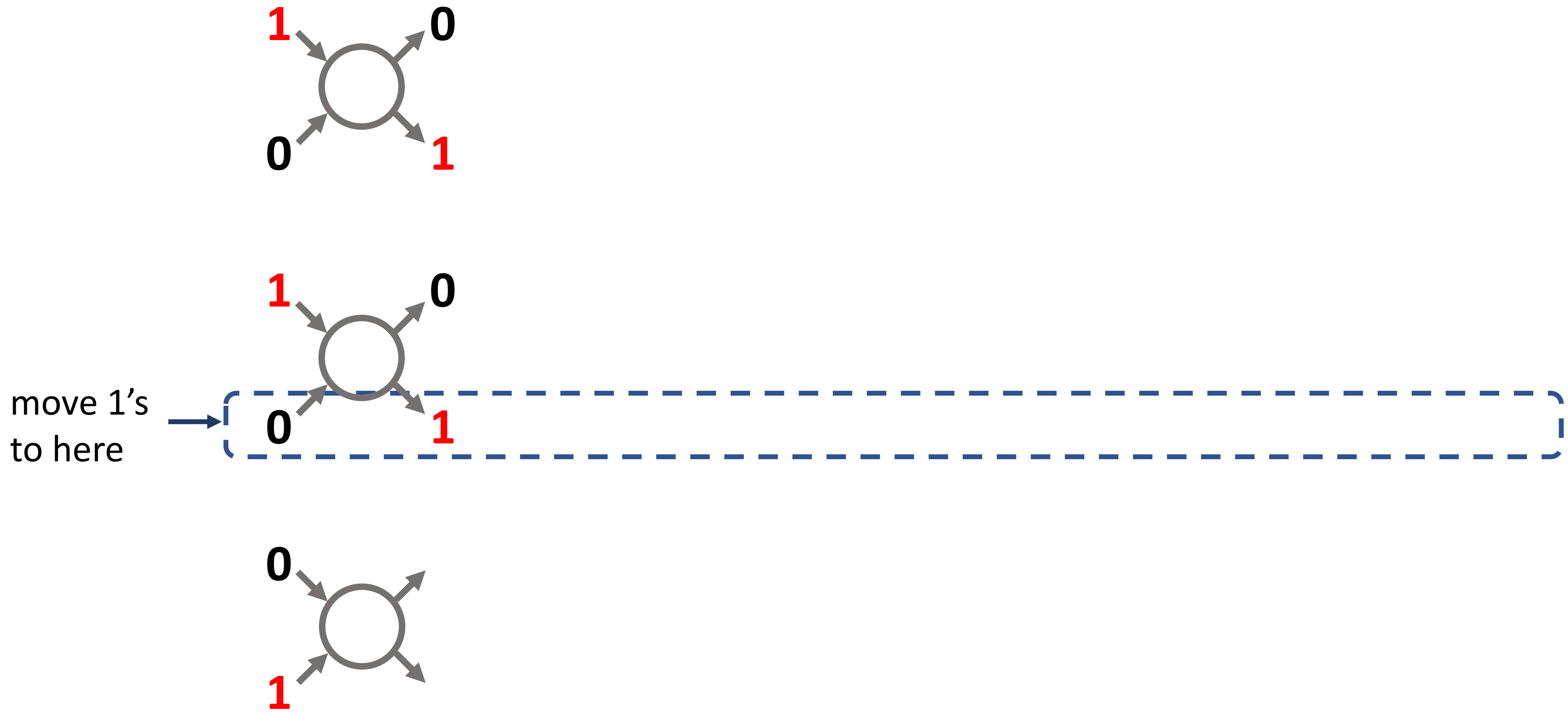
0

1

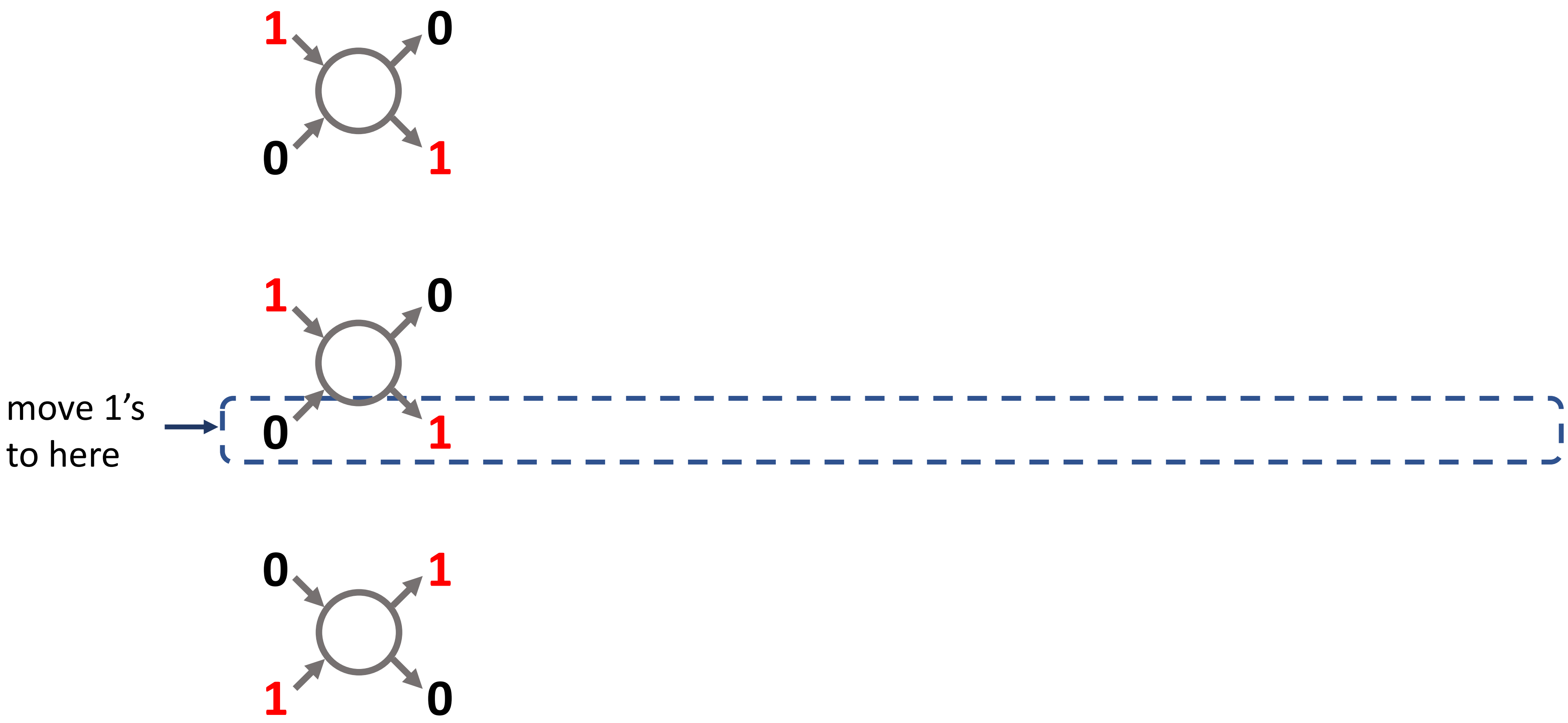
Odd bits



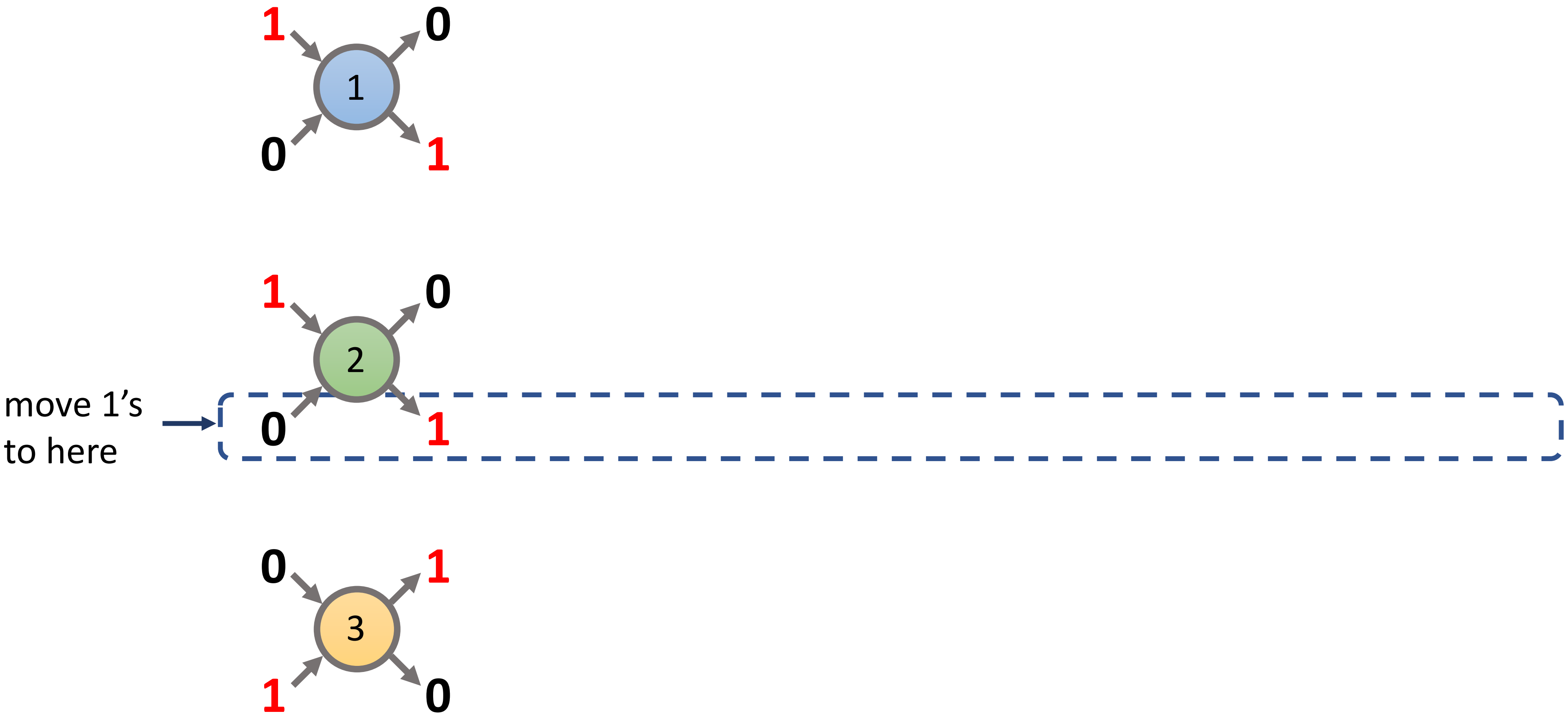
Odd bits



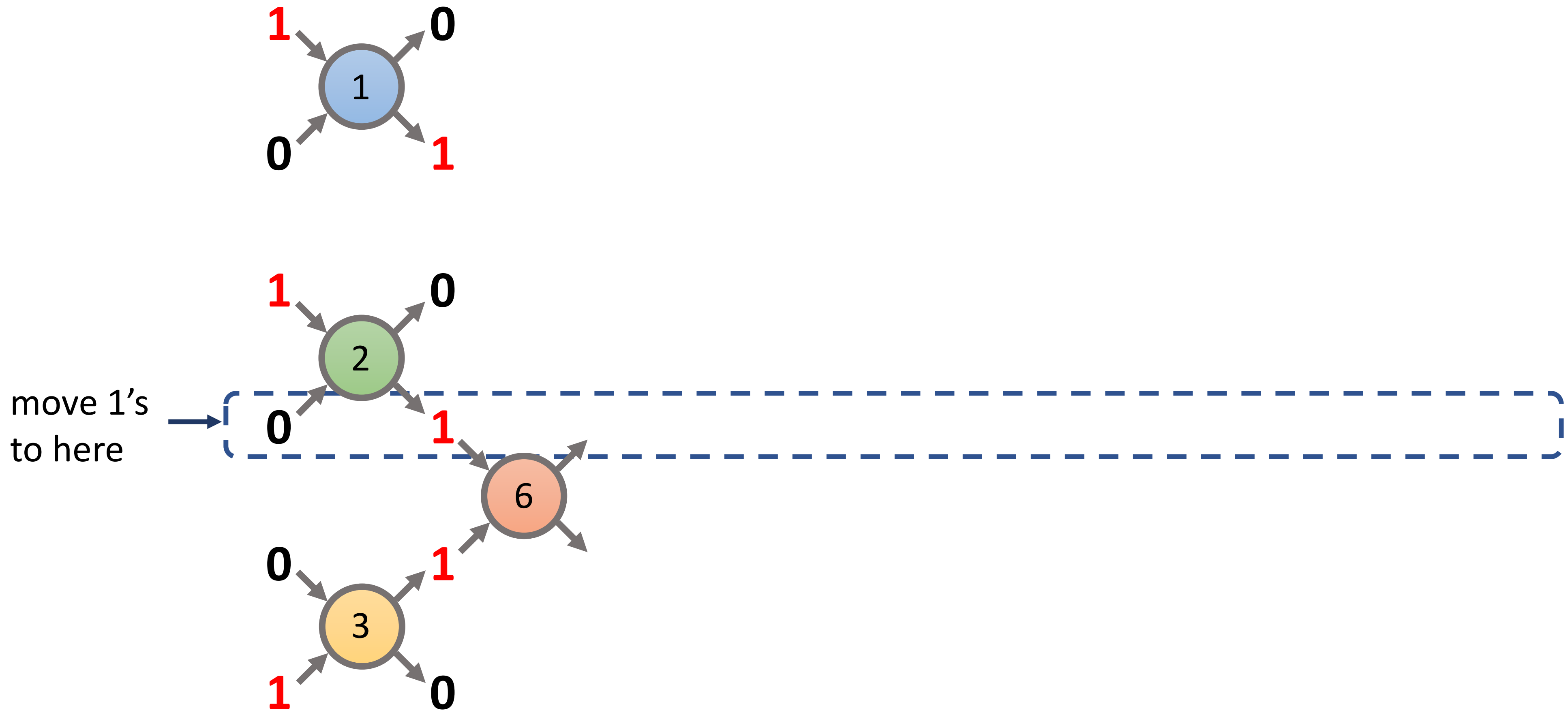
Odd bits



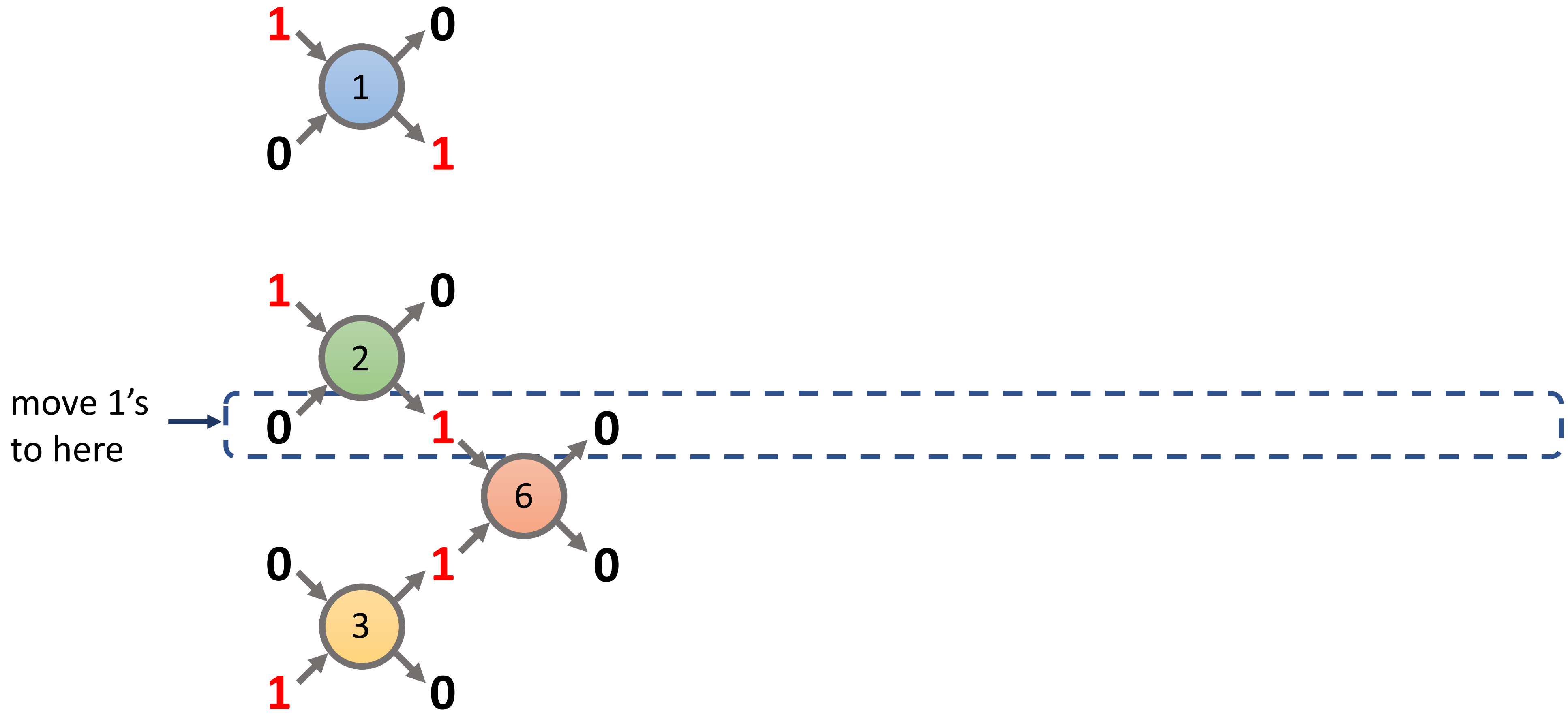
Odd bits



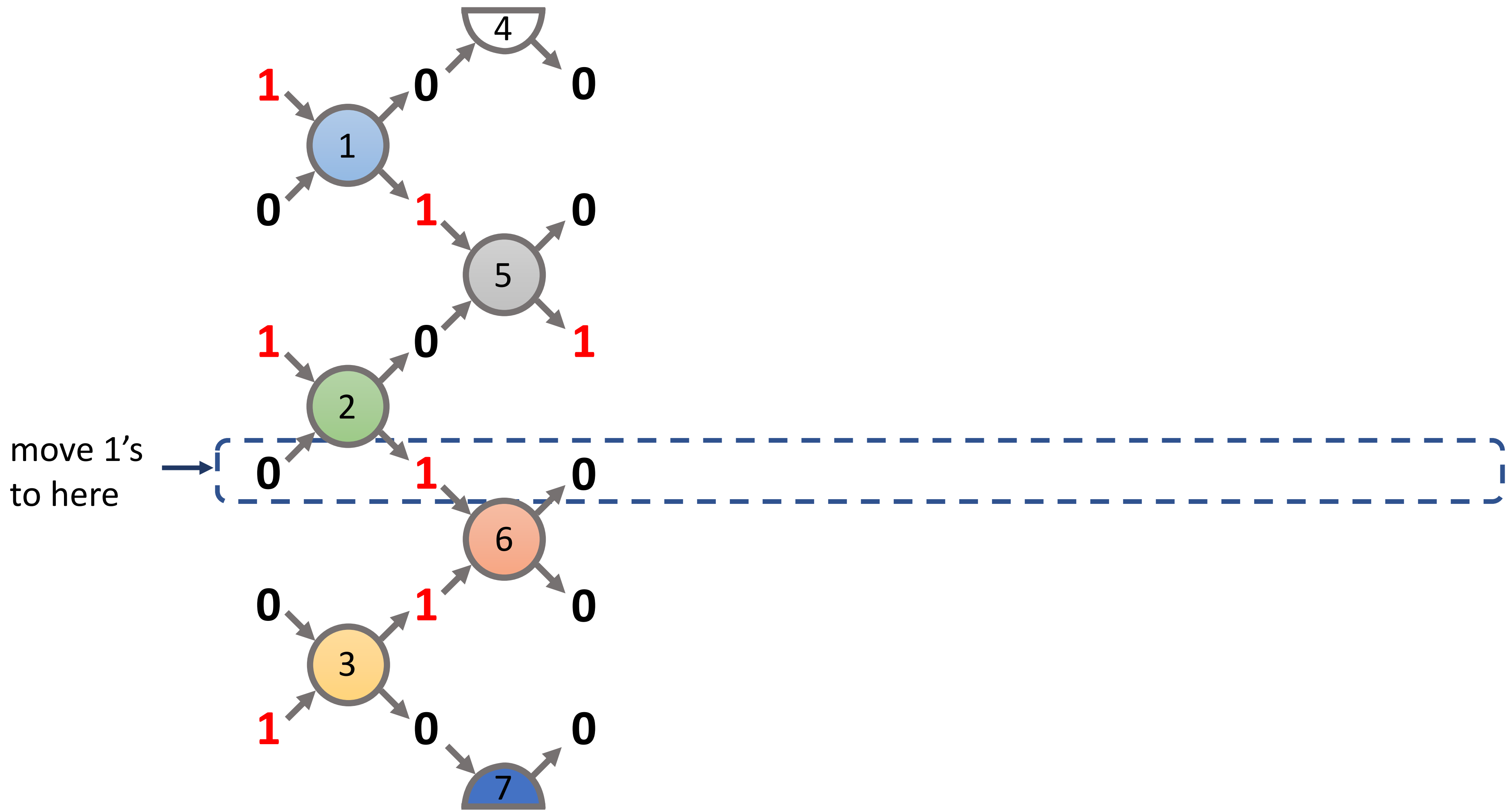
Odd bits



Odd bits

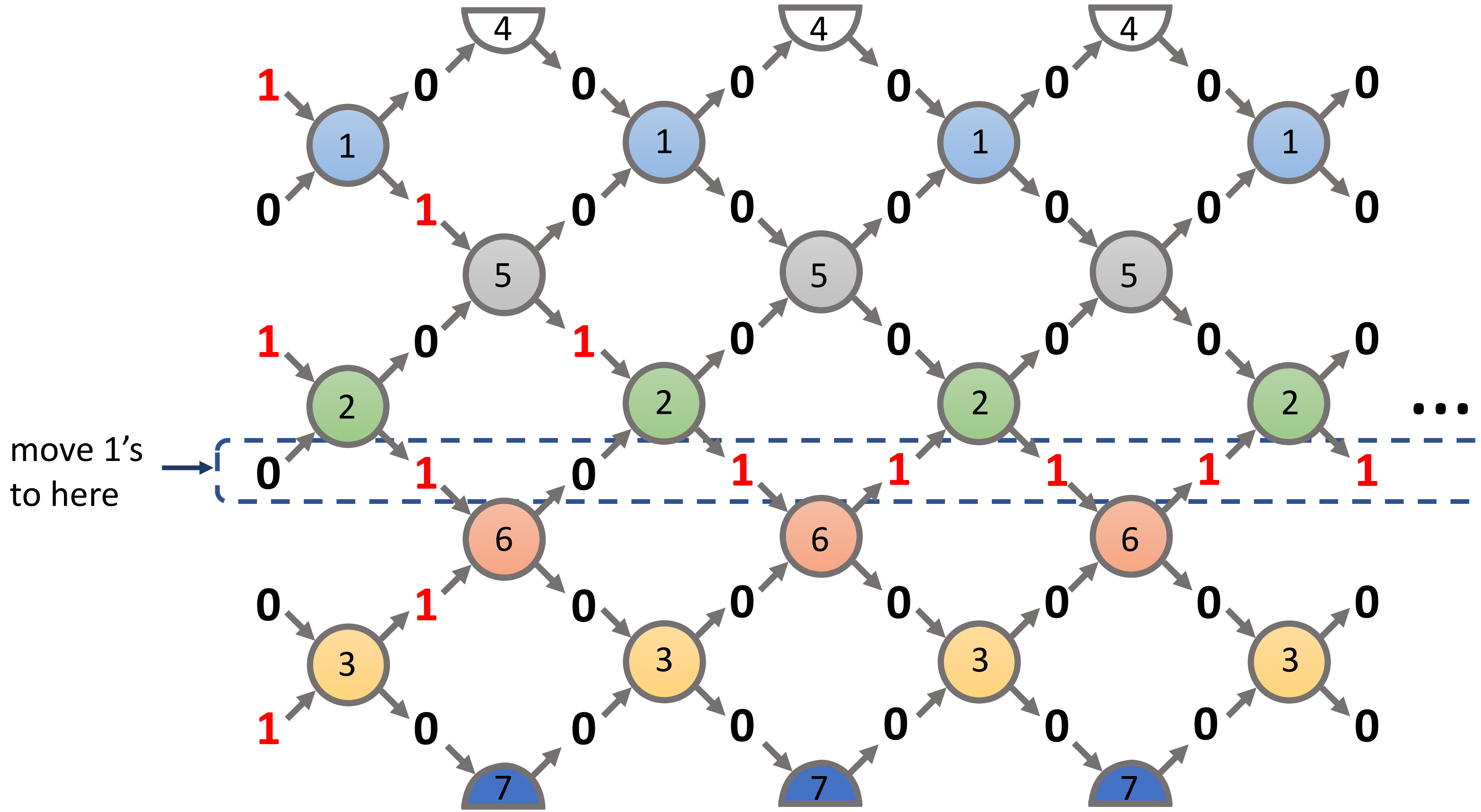


Odd bits



Odd bits

a.k.a. parity



Parity

1

0

1

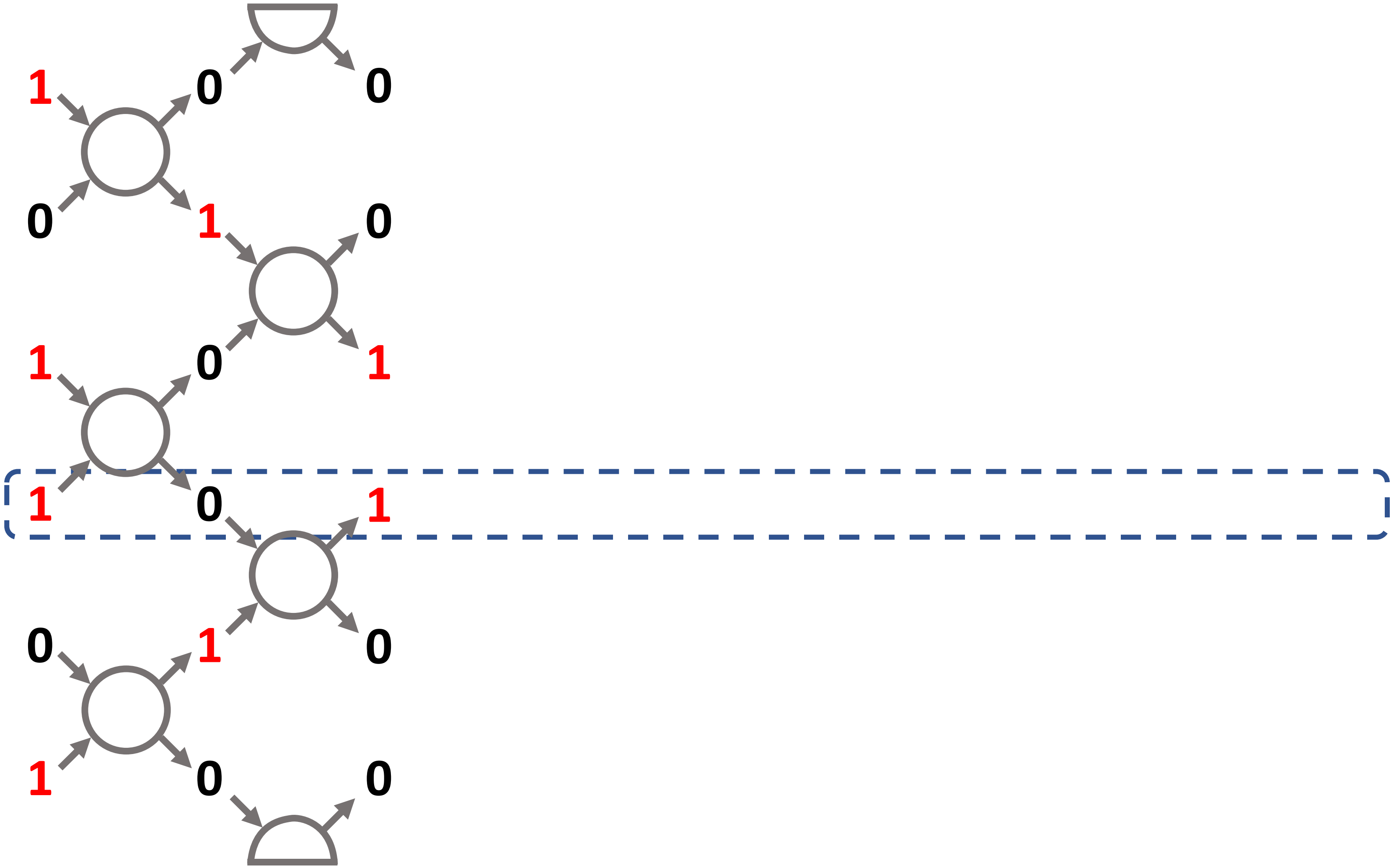
1

0

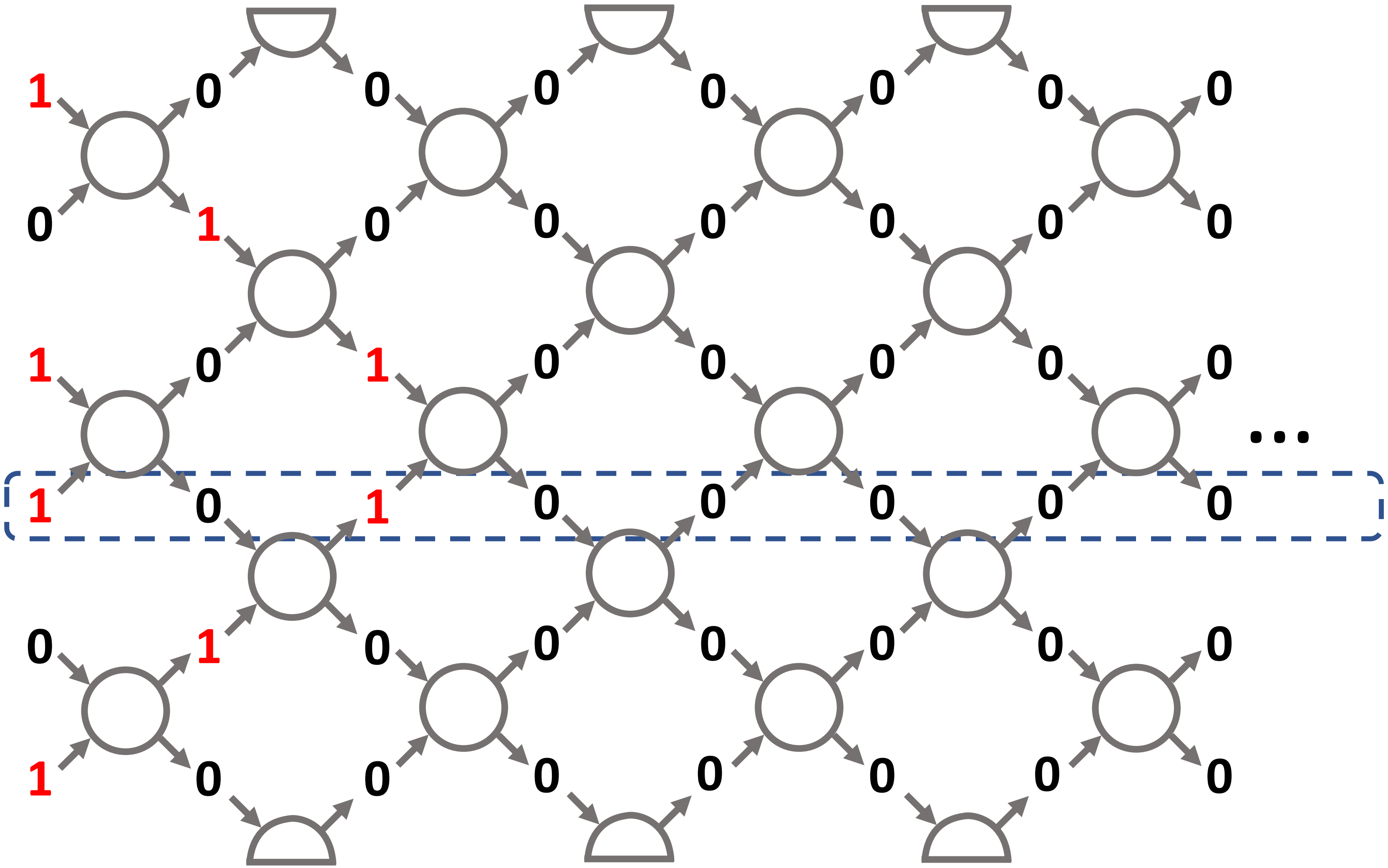
1

Parity

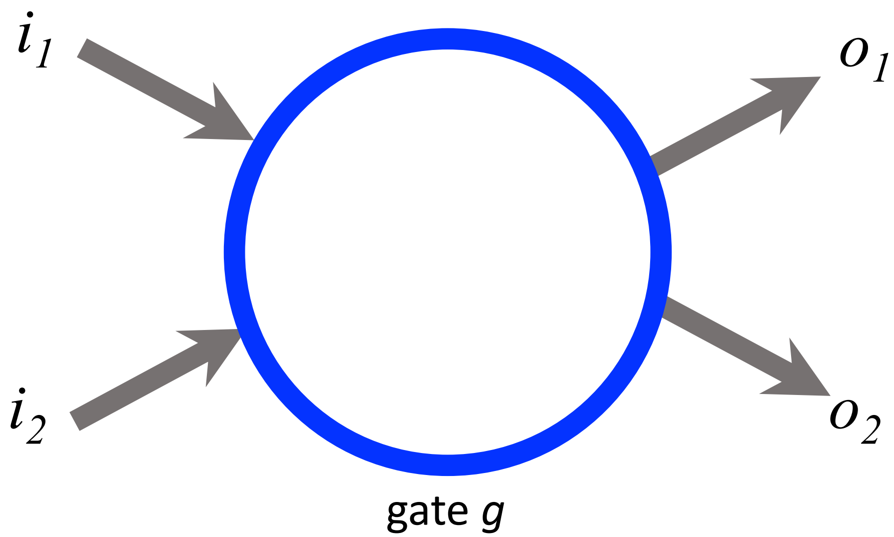
Parity



Parity

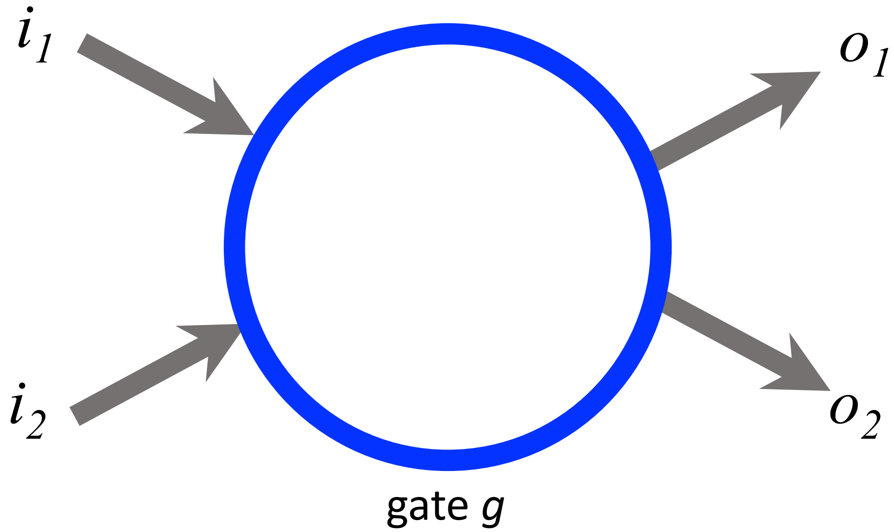


Boolean circuit model



gate g**:** function with two input bits i_1,i_2
and <u>two</u> output bits o_1,o_2

Boolean circuit model

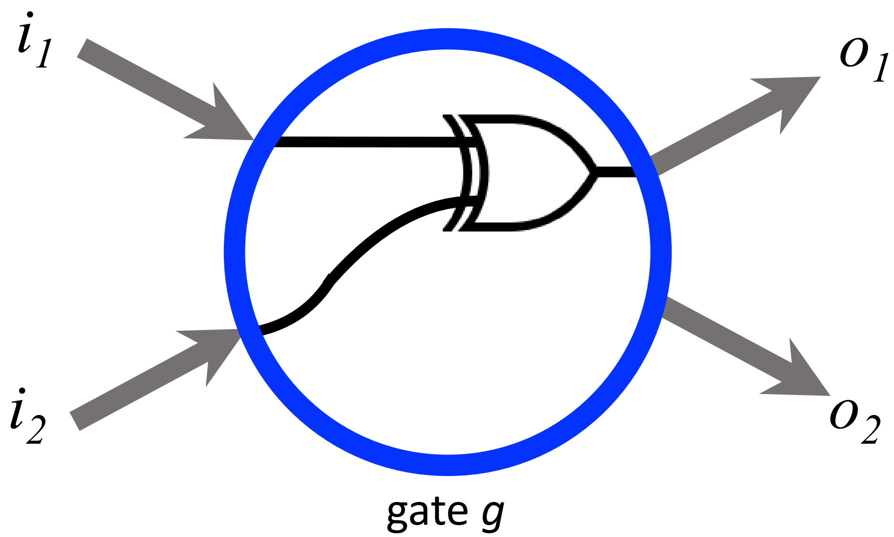


g truth table

i_1	i_2	o_1	o_2
0	0		
0	1		
1	0		
1	1		

gate ***g*****:** function with two input bits i_1,i_2
and two output bits o_1,o_2

Boolean circuit model

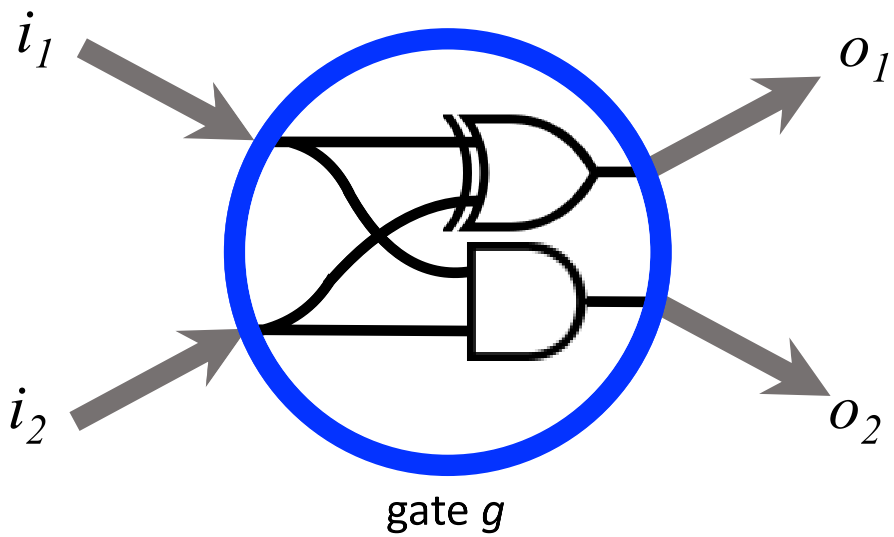


g truth table

i_1	i_2	o_1	o_2
0	0	0	
0	1	1	
1	0	1	
1	1	0	

gate ***g*****:** function with two input bits i_1, i_2
and two output bits o_1, o_2

Boolean circuit model

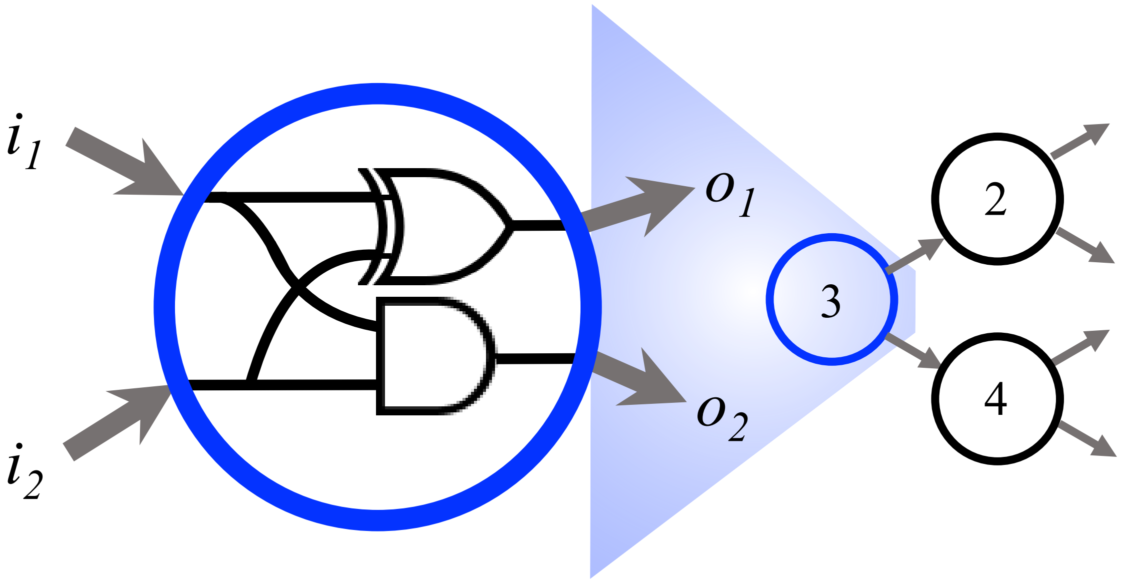


g truth table

i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

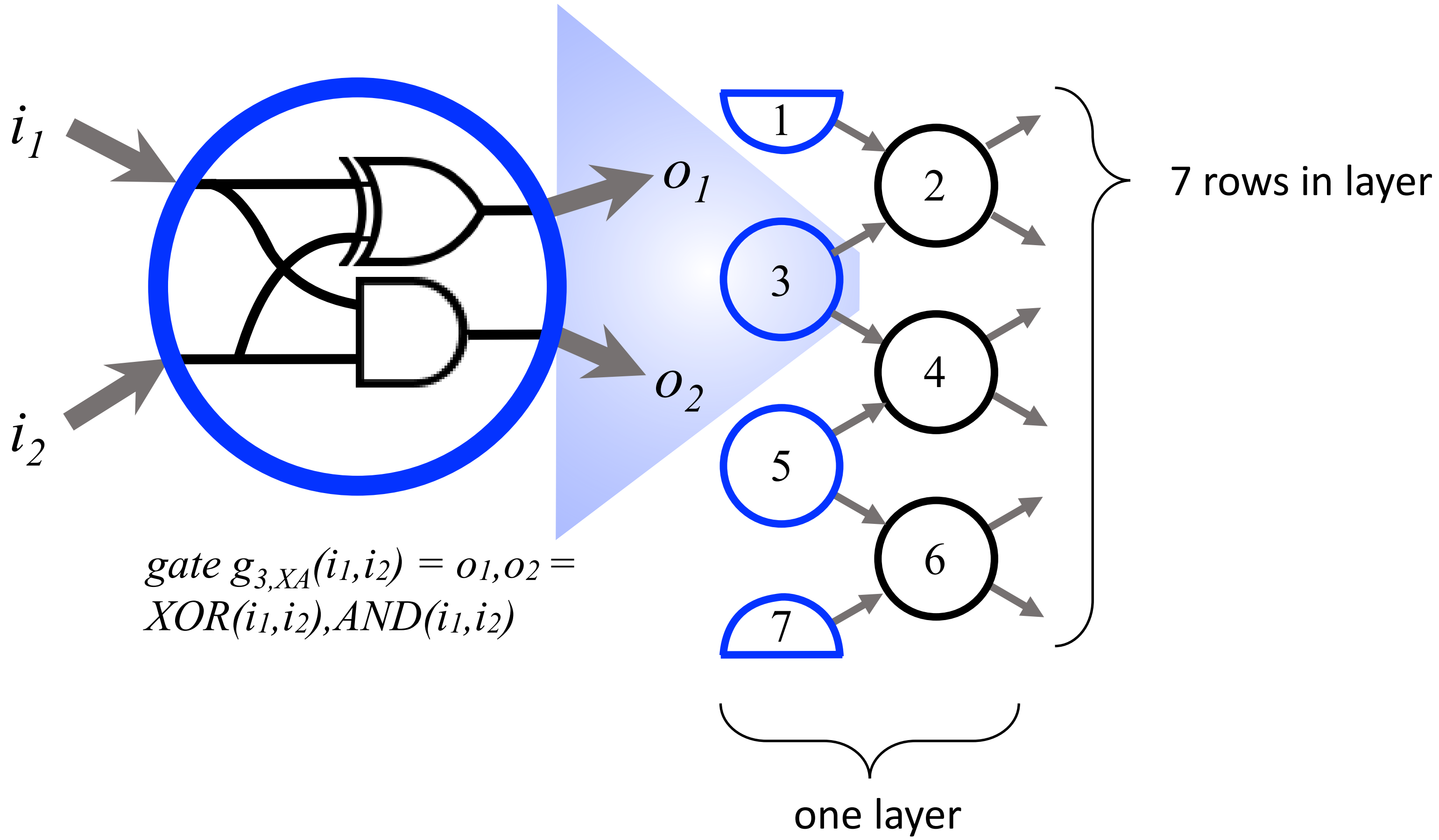
gate ***g*****:** function with two input bits i_1,i_2
and two output bits o_1,o_2

Boolean circuit model



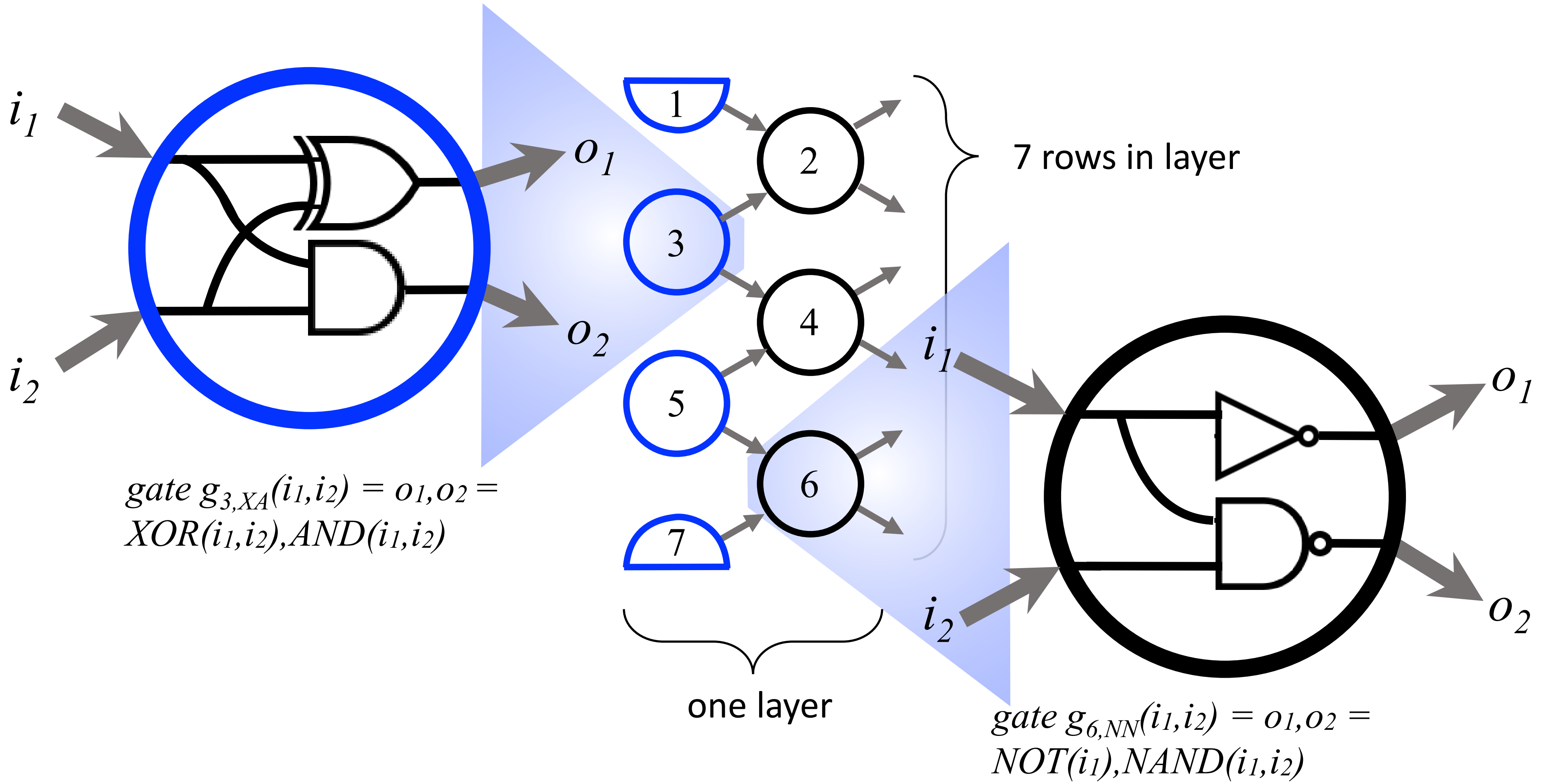
gate $g_{3,XA}(i_1,i_2) = o_1,o_2 = XOR(i_1,i_2),AND(i_1,i_2)$

Boolean circuit model



Gates organized into vertical layers consisting of many rows

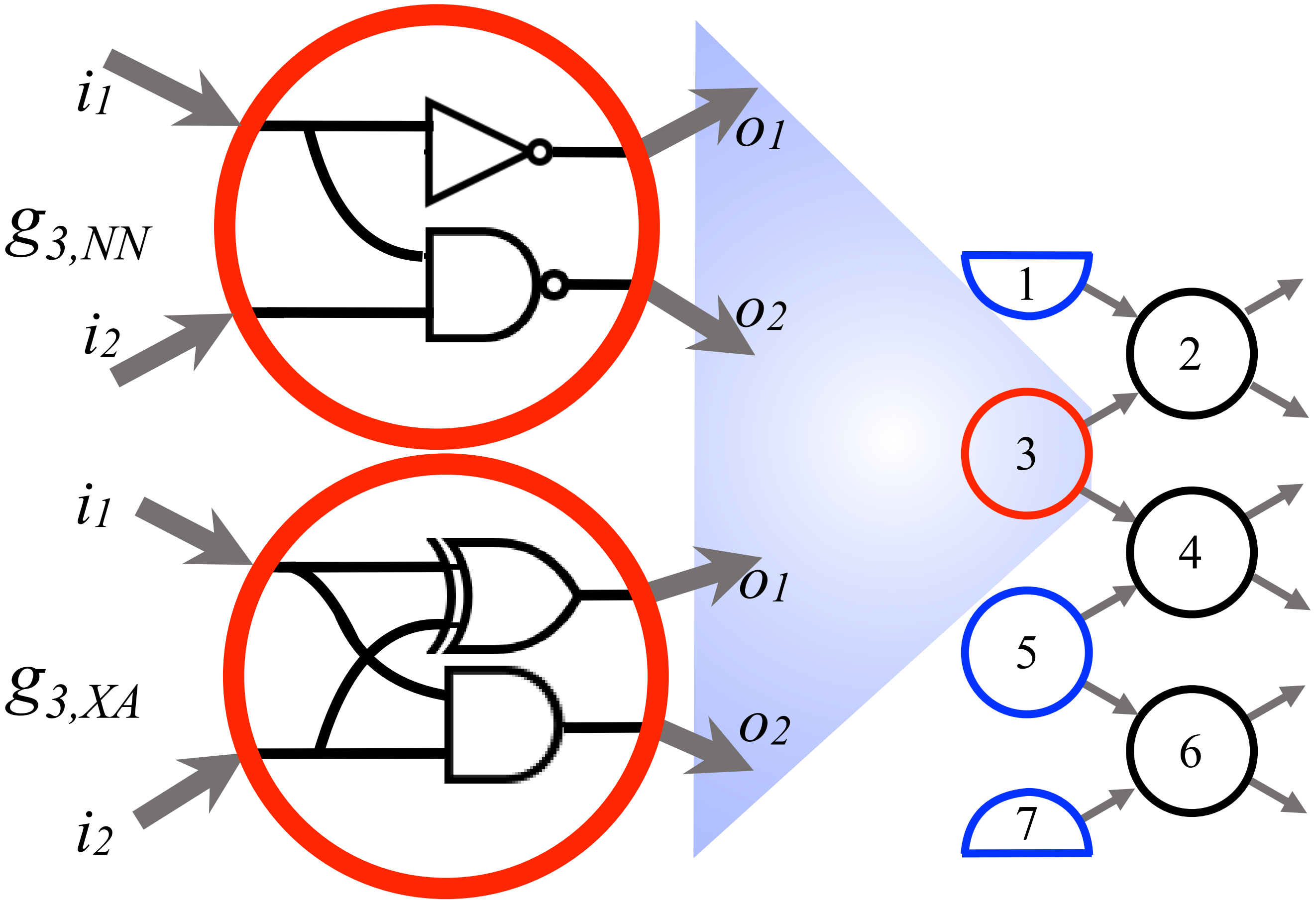
Boolean circuit model



Gates organized into vertical layers consisting of many rows

Possibly different gate types on different rows within a layer

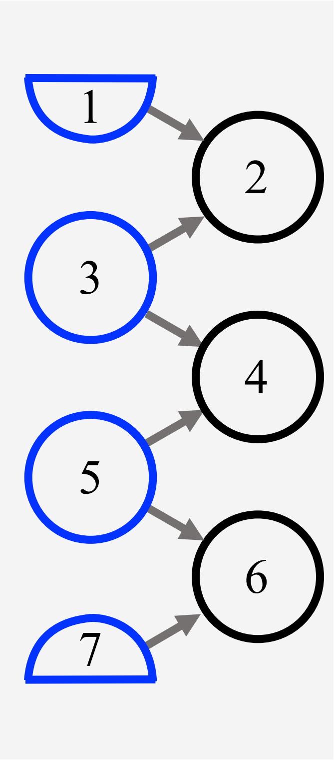
Boolean circuit model



Randomization: Each row may be assigned ≥ 2 gates, with associated probabilities, e.g., $\Pr[g_{3,NN}] = \Pr[g_{3,XA}] = ½$

Boolean circuit model

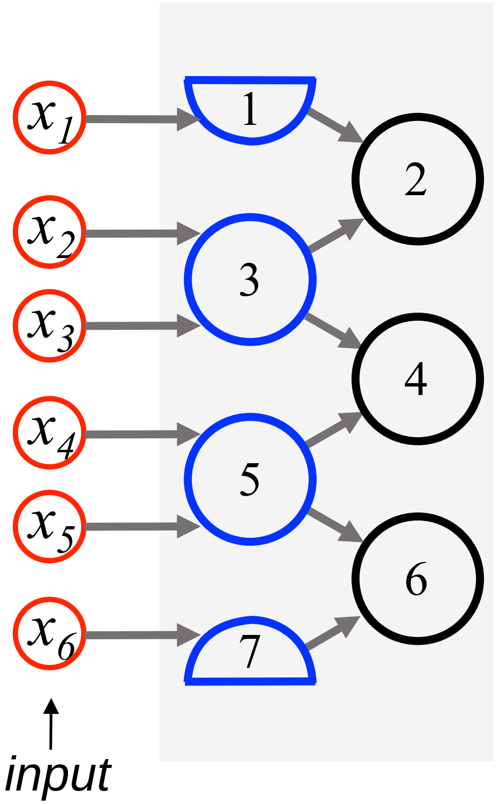
Programmer specifies layer: gates to go in each row



Boolean circuit model

Programmer specifies layer: gates to go in each row

User gives n input bits $x \in \{0,1\}^n$

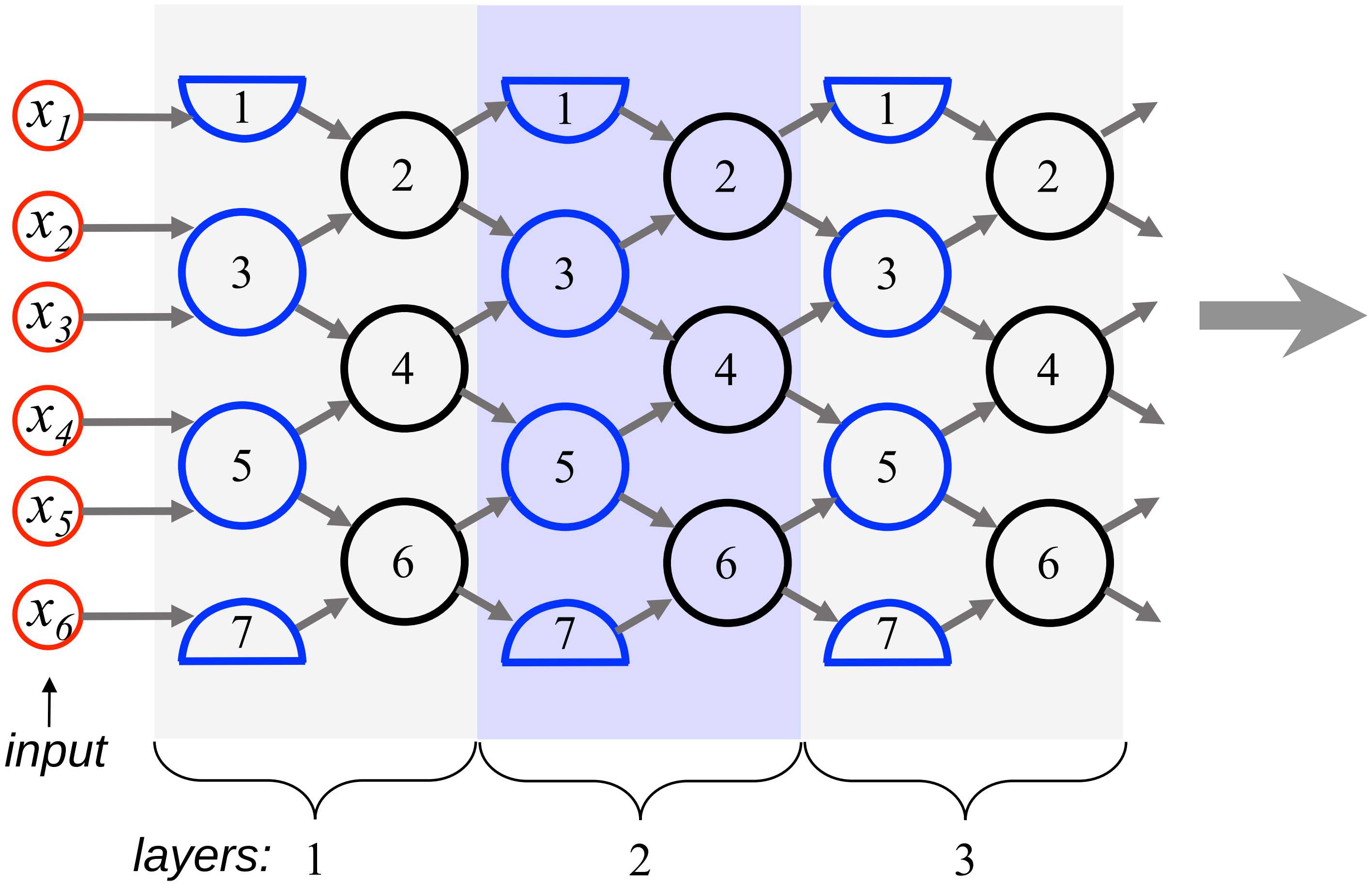


Boolean circuit model

Programmer specifies layer: gates to go in each row

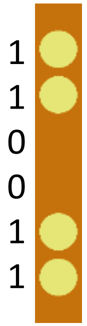
User gives n input bits $x \in \{0,1\}^n$

Computation flows from inputs to layers 1→2→3→…

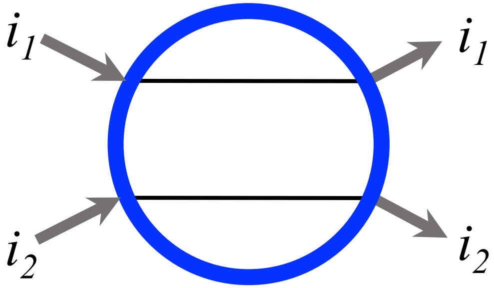


Example circuits with same gate in every row

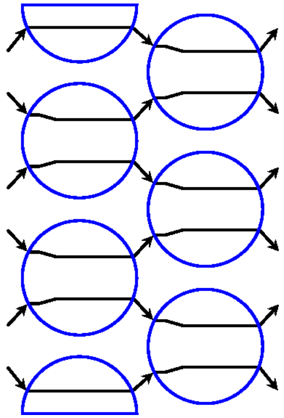
COPY



COPY gates

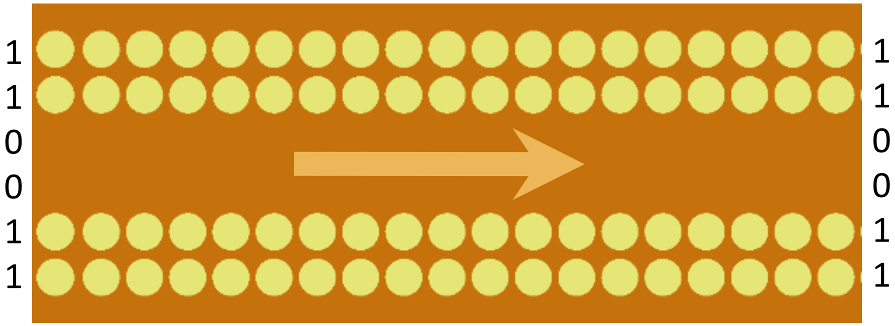


i_1	i_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

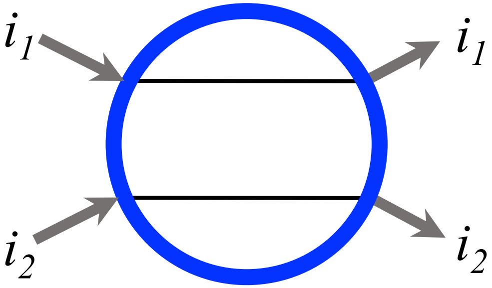


Example circuits with same gate in every row

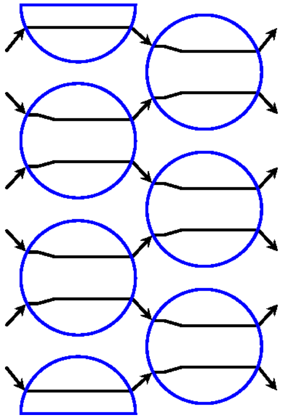
COPY



COPY gates

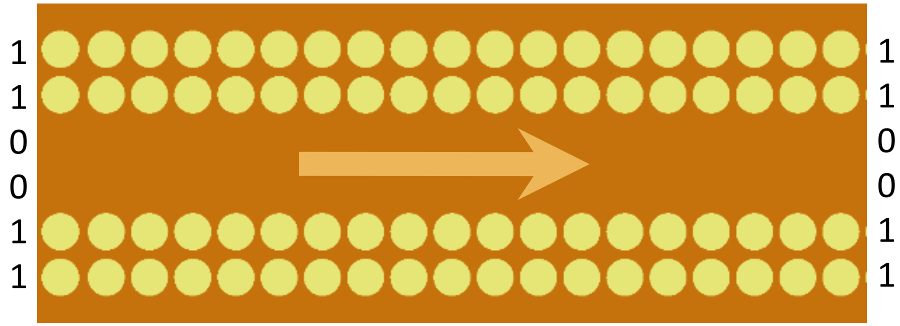


i_1	i_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

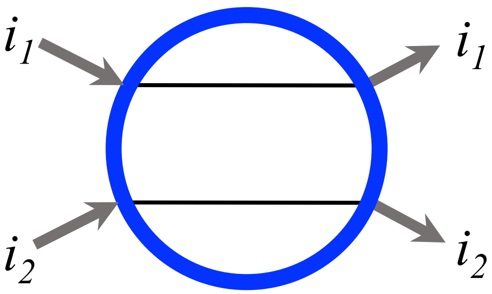


Example circuits with same gate in every row

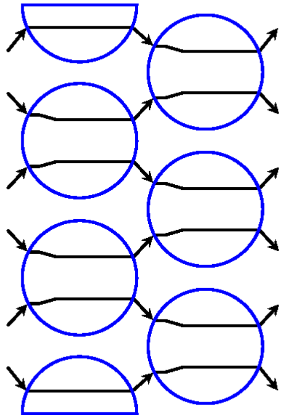
COPY



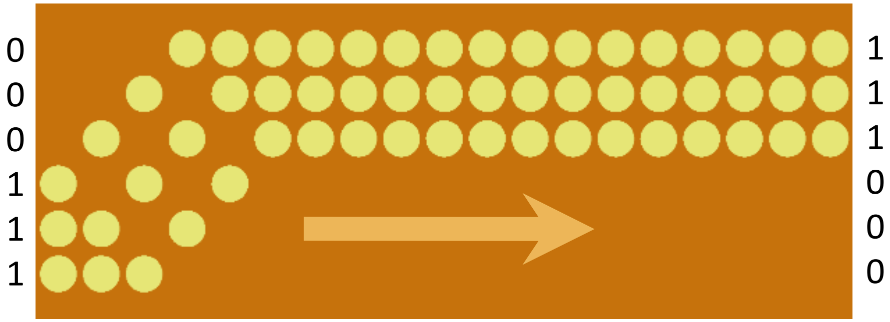
COPY gates



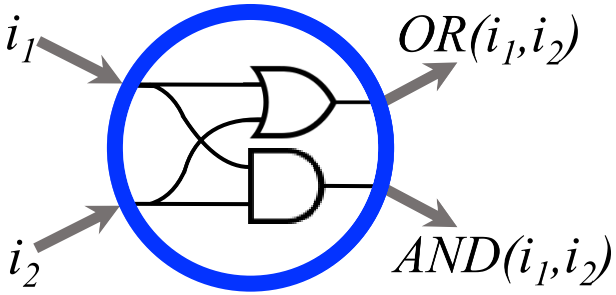
i_1	i_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1



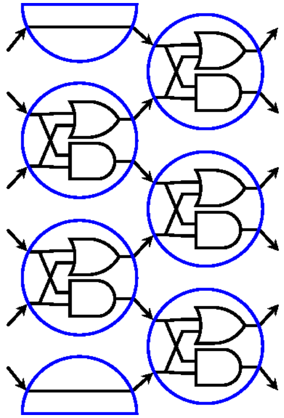
SORTING



SORTING gates

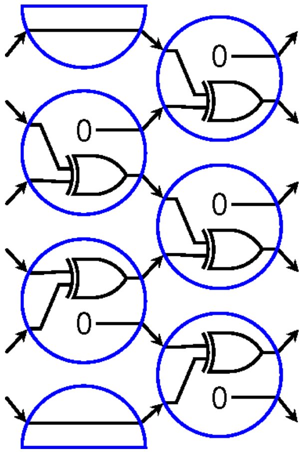


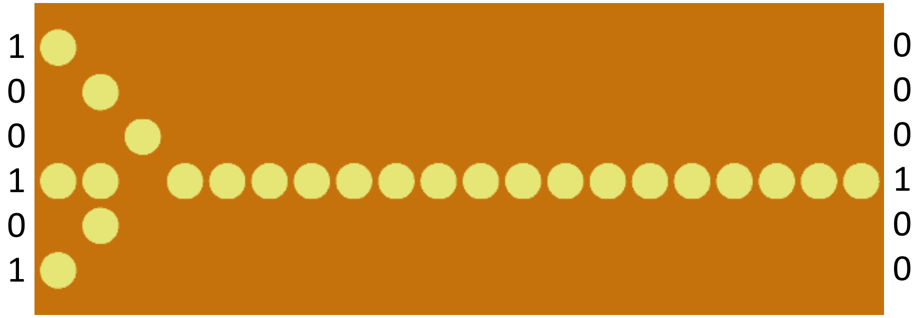
i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1



Example circuits with different gates in each row

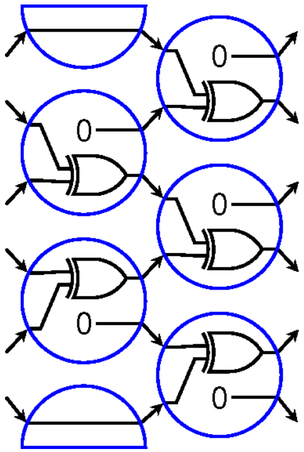
PARITY

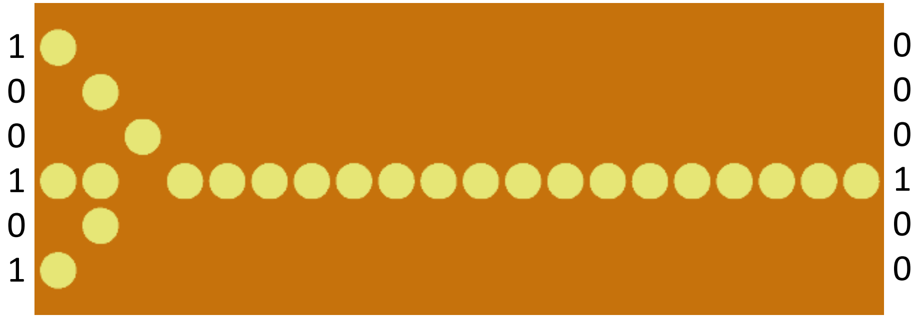




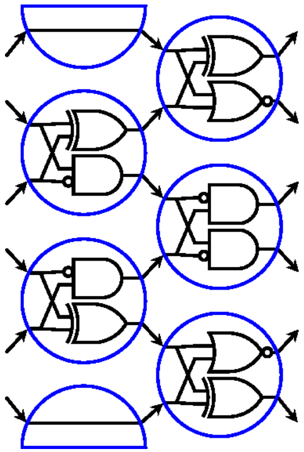
Example circuits with different gates in each row

PARITY

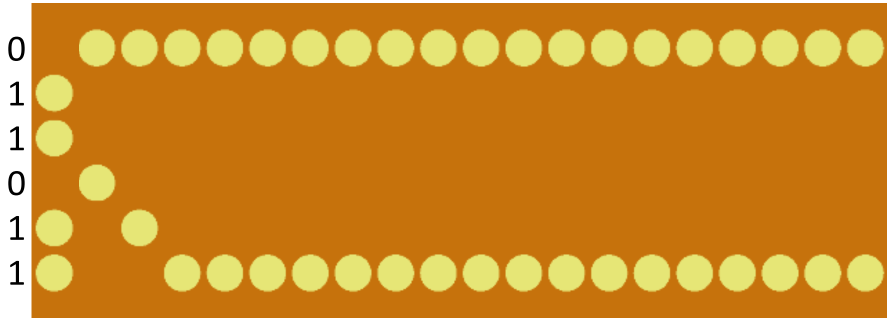




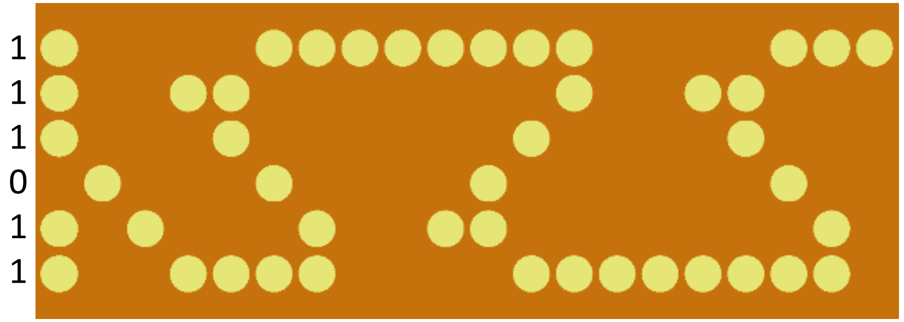
MULTIPLEOF3



$011011_2 = 27_{10} = 3 \cdot 9$



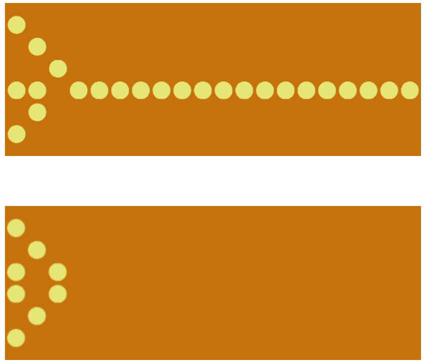
$111011_2 = 59_{10} = 3 \cdot 19 + 2$



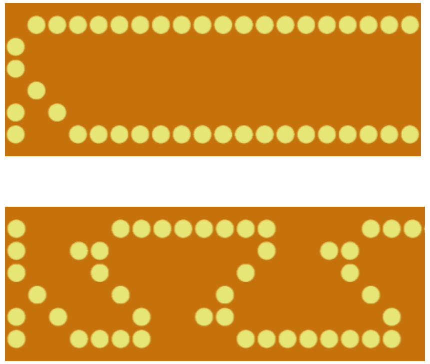
Deterministic circuits

answer yes/no question

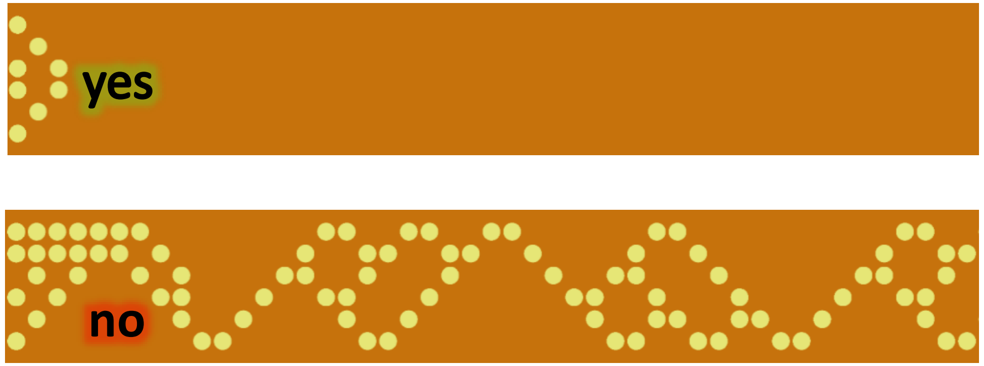
PARITY



MULTIPLEOF3



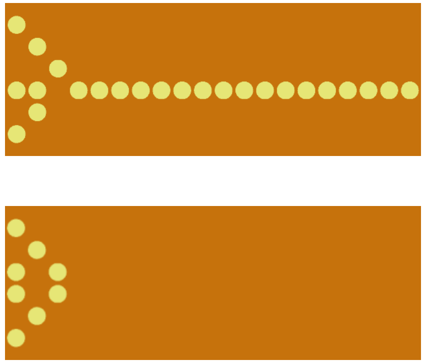
PALINDROME



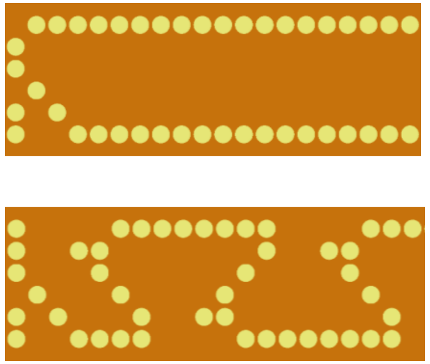
Deterministic circuits

answer yes/no question

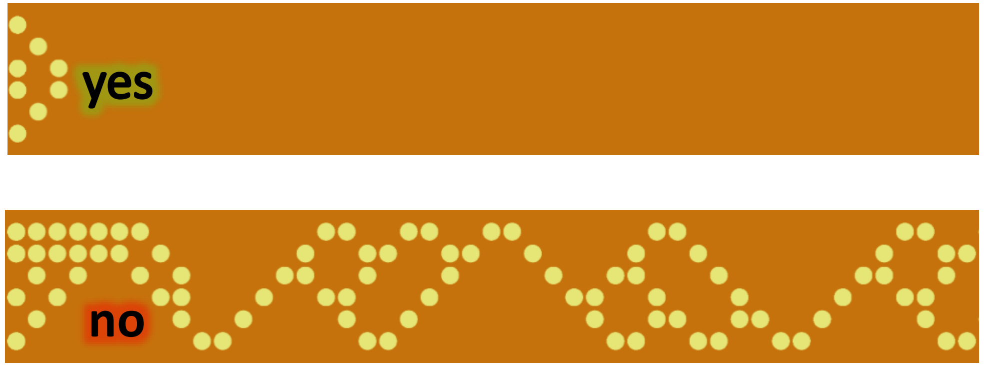
PARITY



MULTIPLEOF3

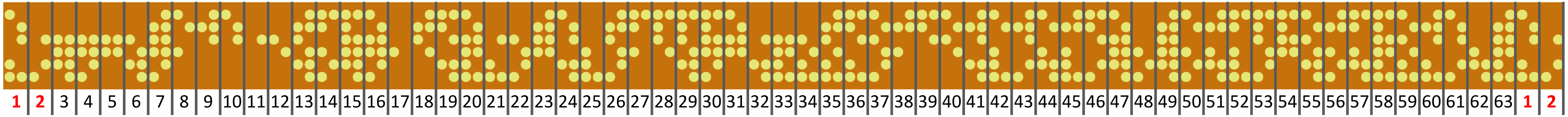


PALINDROME



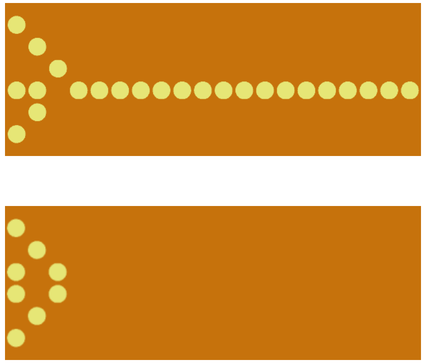
CYCLE63

"count" as high as possible

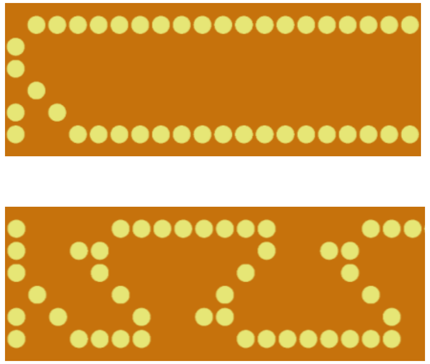


Deterministic circuits

PARITY

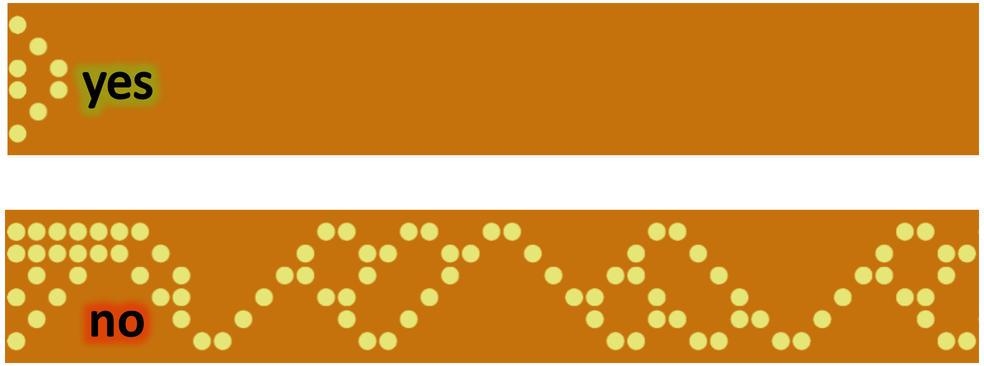


MULTIPLEOF3



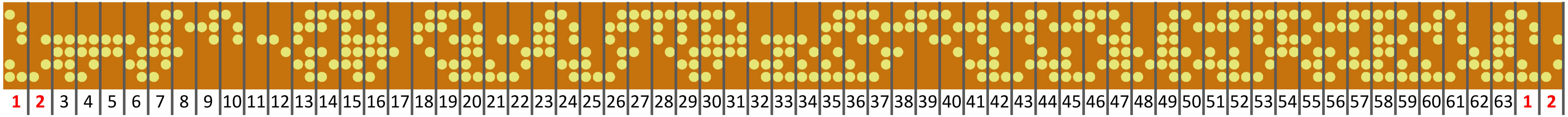
PALINDROME

answer yes/no question



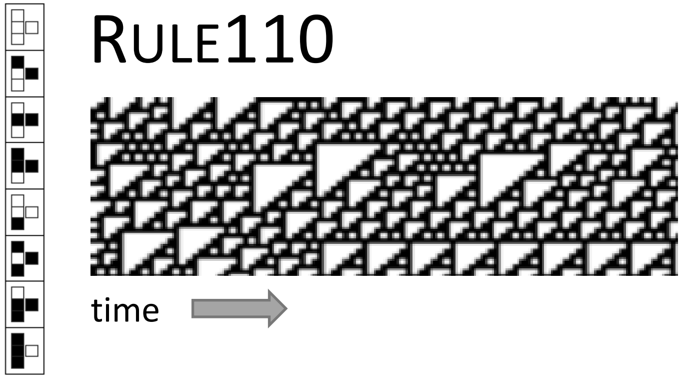
CYCLE63

"count" as high as possible



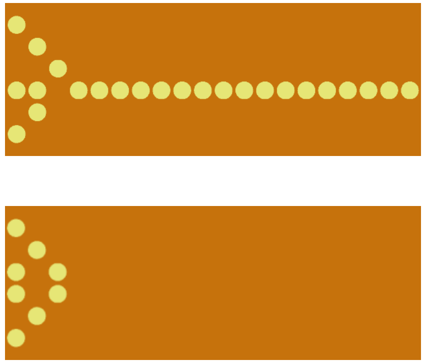
RULE110

simulate cellular automata

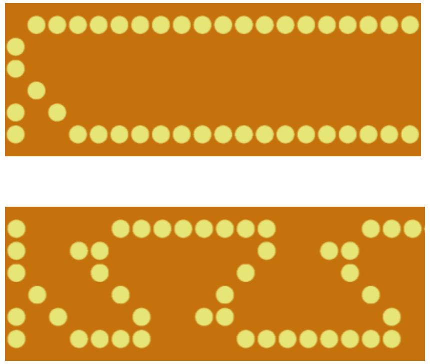


Deterministic circuits

PARITY

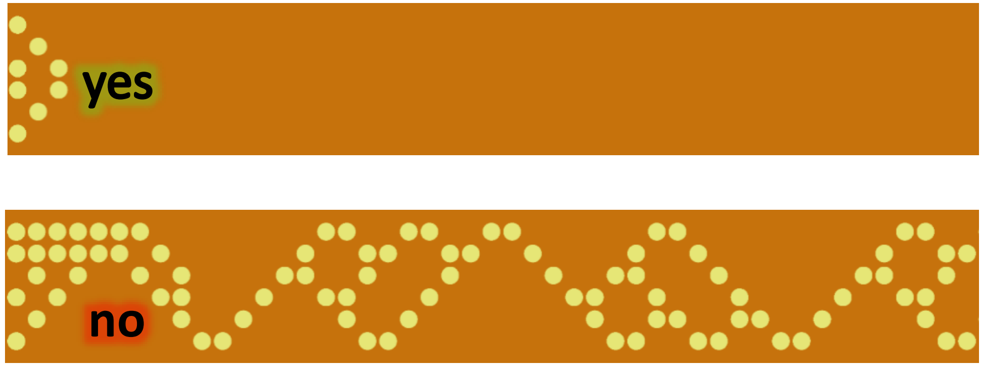


MULTIPLEOF3



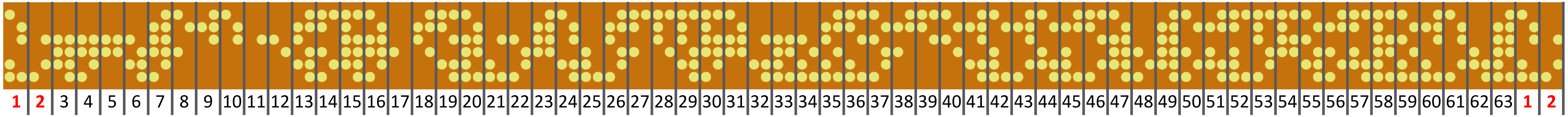
PALINDROME

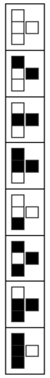
answer yes/no question



CYCLE63

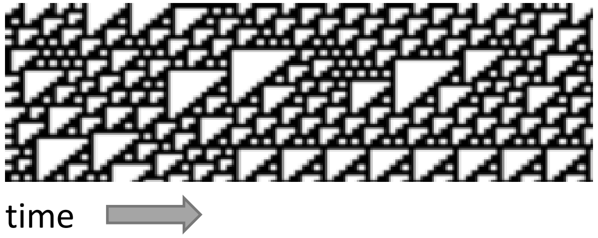
"count" as high as possible

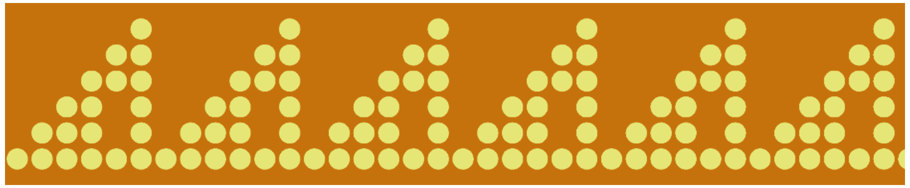




RULE110

simulate cellular automata





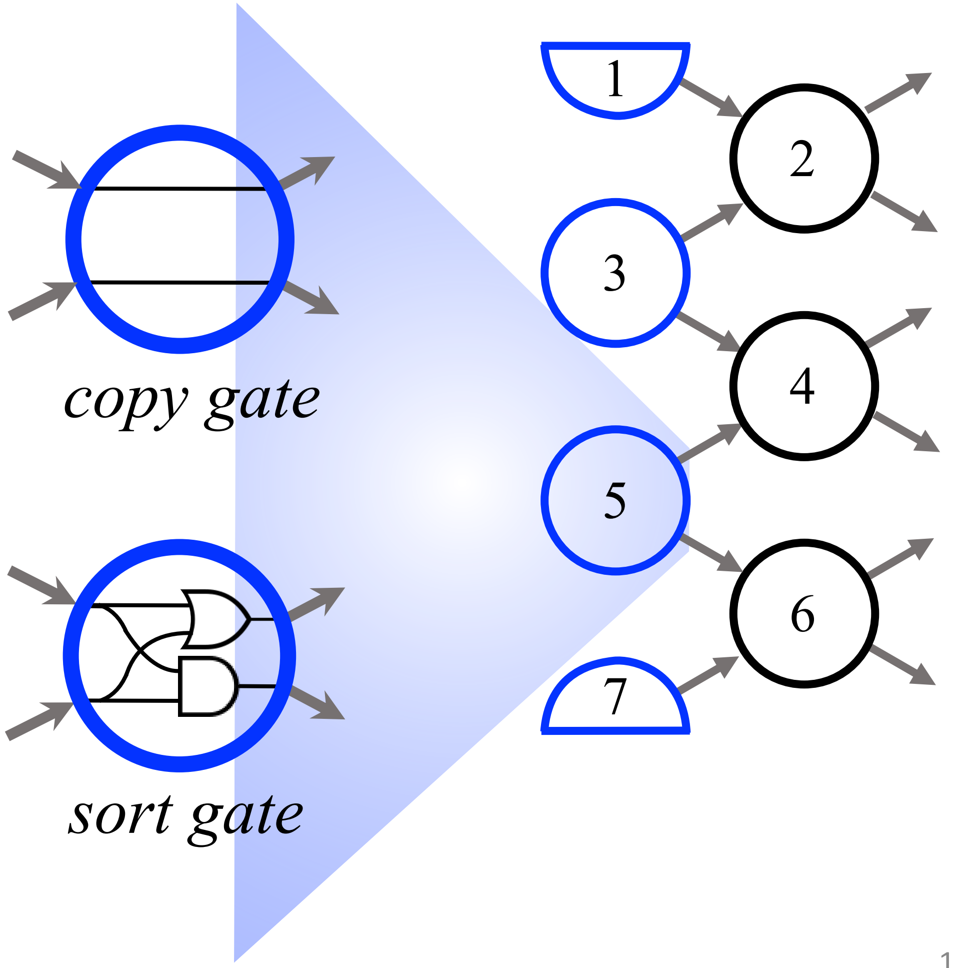
Theorem: Rule 110 can efficiently execute any algorithm.

[Cook, Complex Systems 2004]

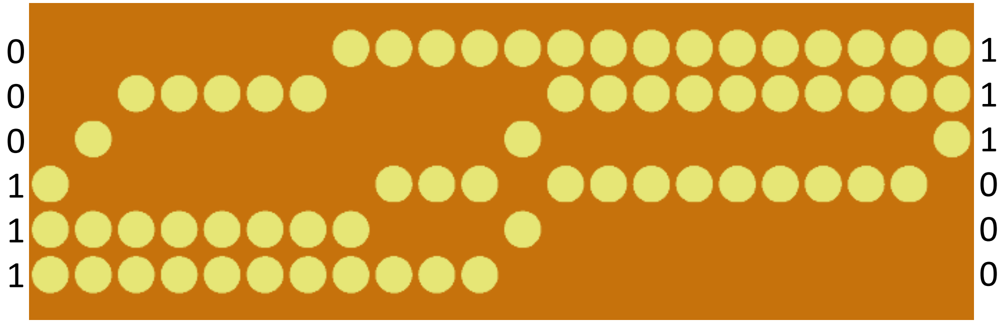
[Neary, Woods, ICALP 2006]

Randomization: “Lazy” sorting

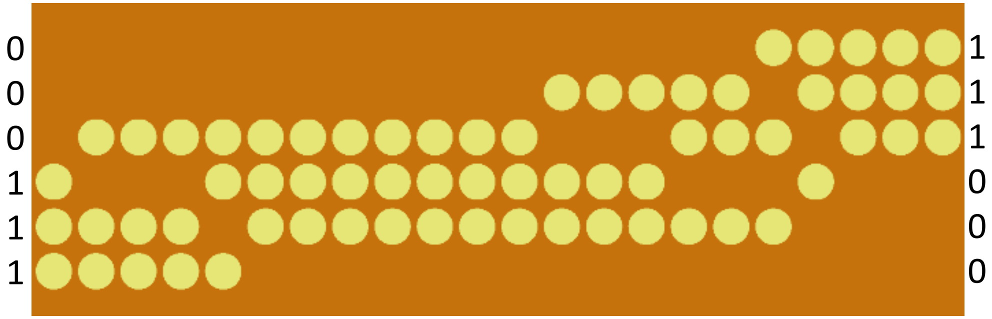
If 1 and 0 out of order, flip a coin to decide whether to swap them.

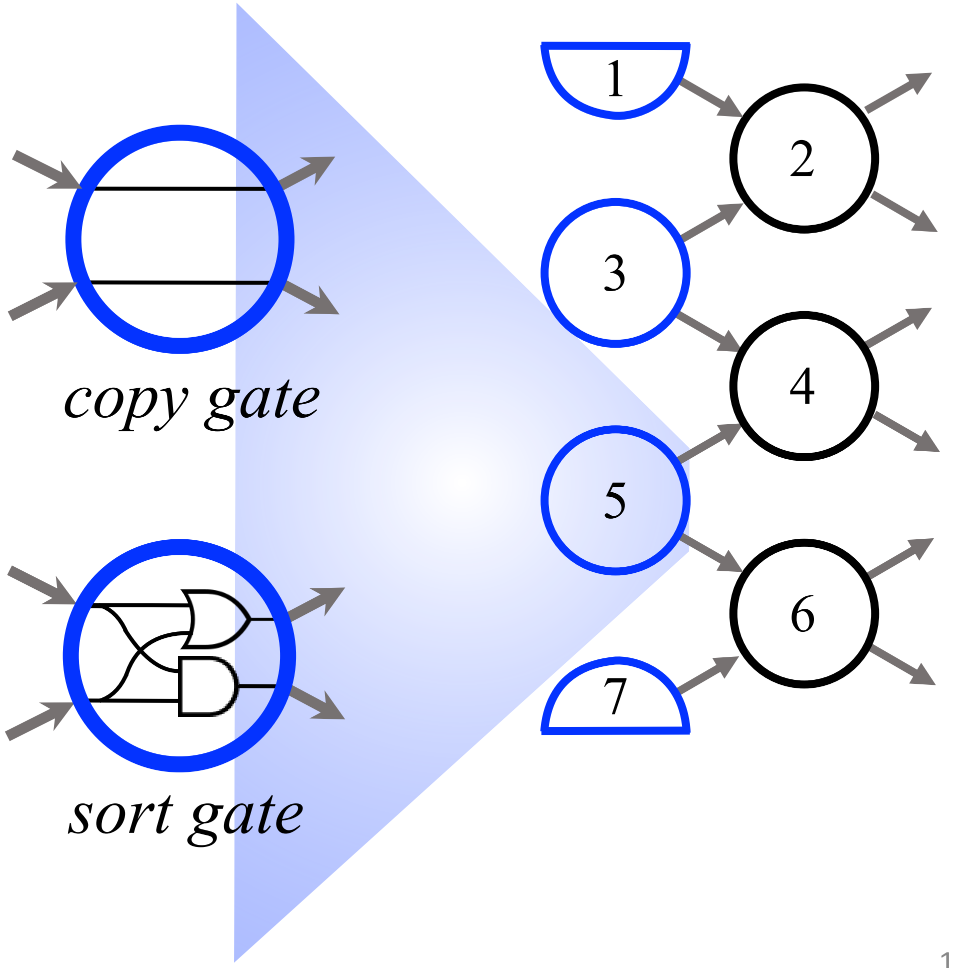


Randomization: "Lazy" sorting



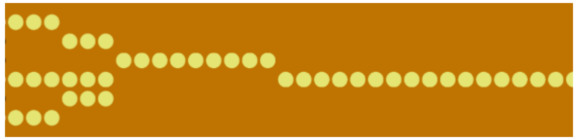
If 1 and 0 out of order, flip a coin to decide whether to swap them.

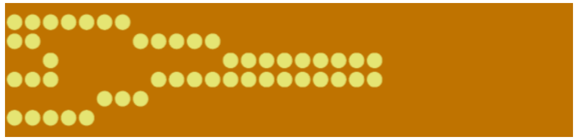




Randomized circuits

LAZYPARITY



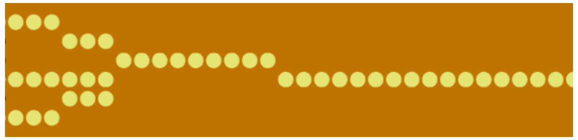


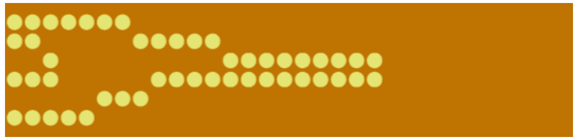
RANDOMWALKINGBIT

DIAMONDSAREFOREVER

Randomized circuits

LAZYPARITY





RANDOMWALKINGBIT

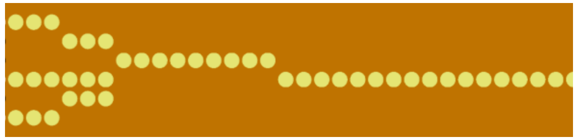
DIAMONDSAREFOREVER

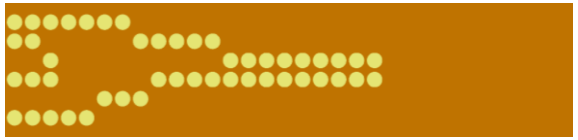
FAIRCOIN

use biased coin to
simulate unbiased coin

Randomized circuits

LazyParity





RandomWalkingBit

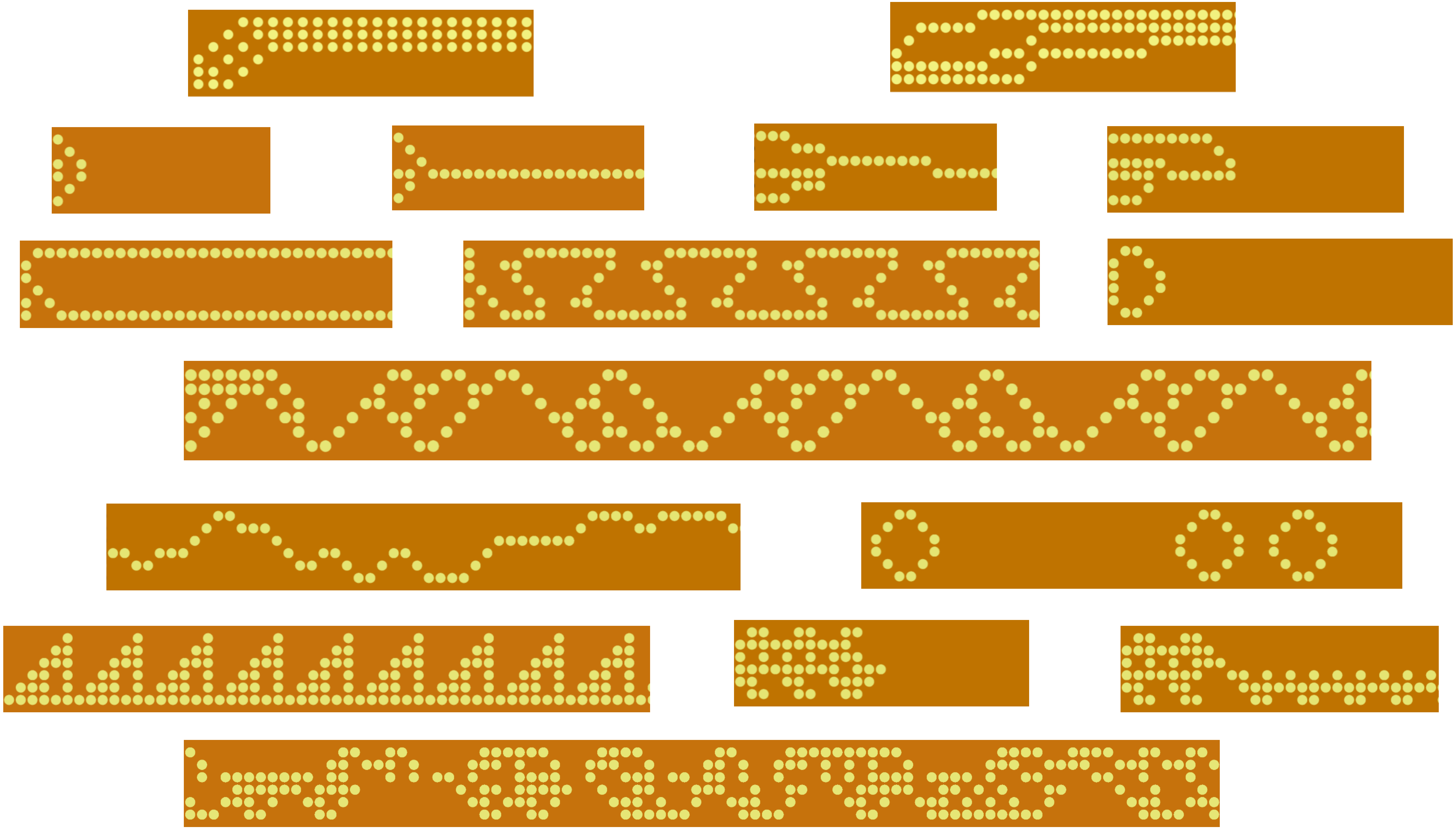
DiamondsAreForever

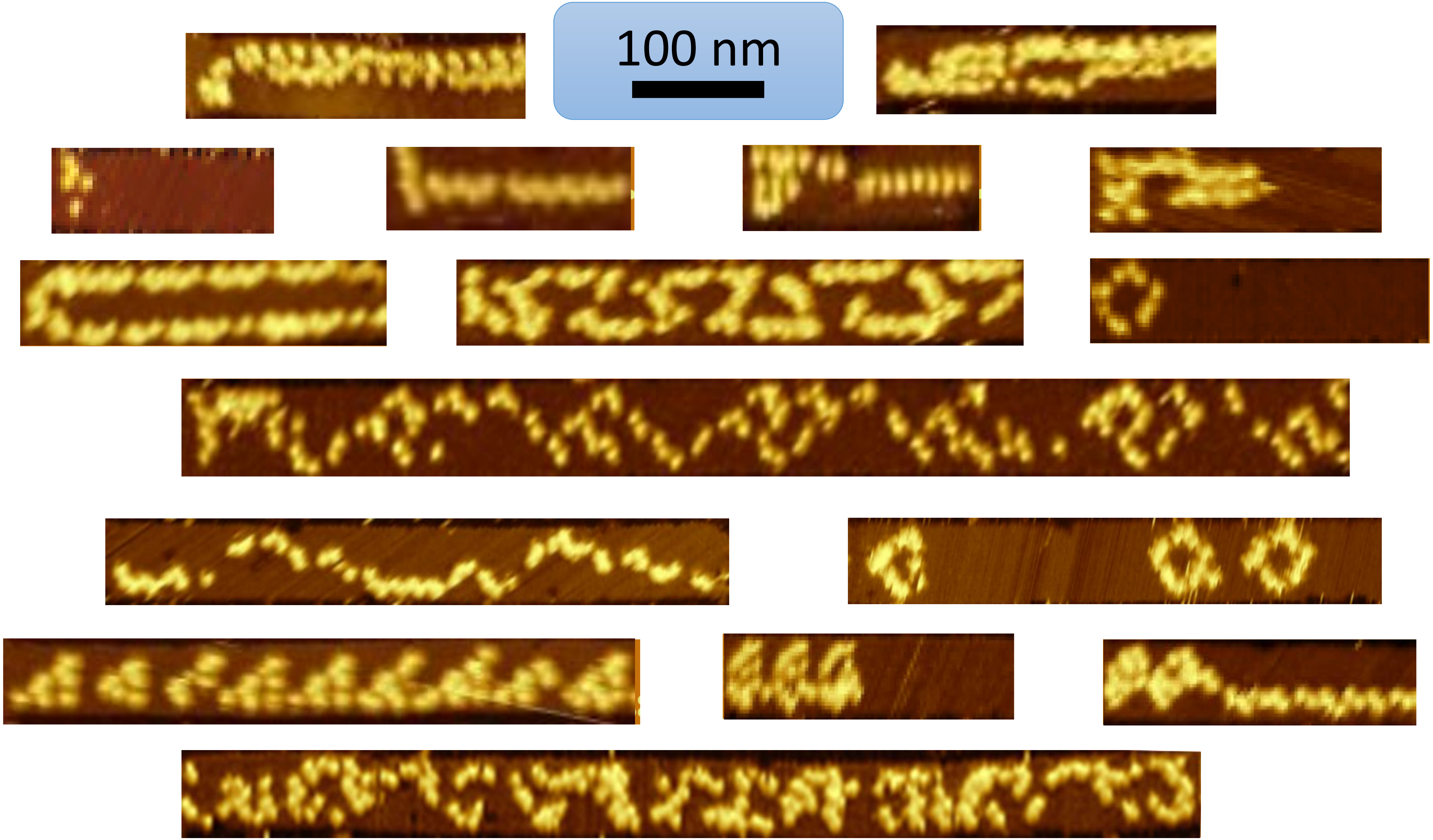
FairCoin

use biased coin to
simulate unbiased coin

Pr [] = Pr [] = ½

for **any** (positive) probabilities for the randomized gate

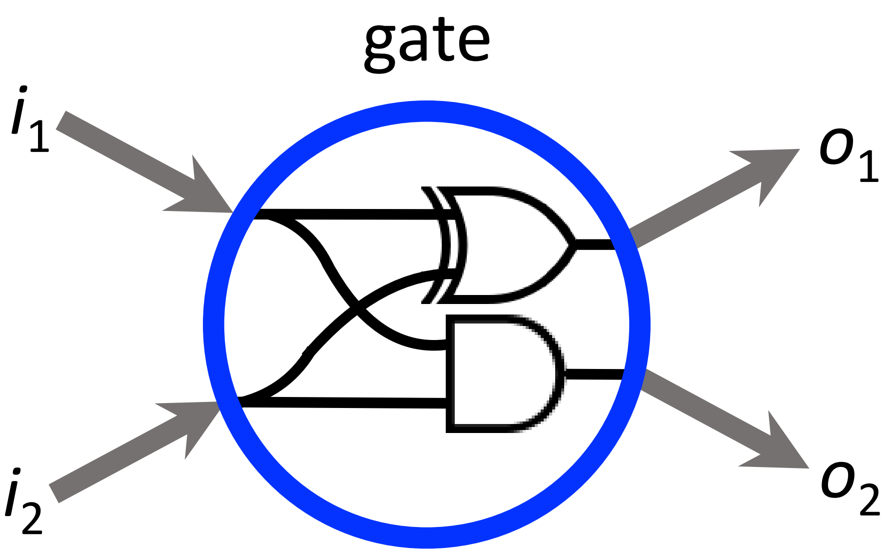



100 nm

Hierarchy of abstractions

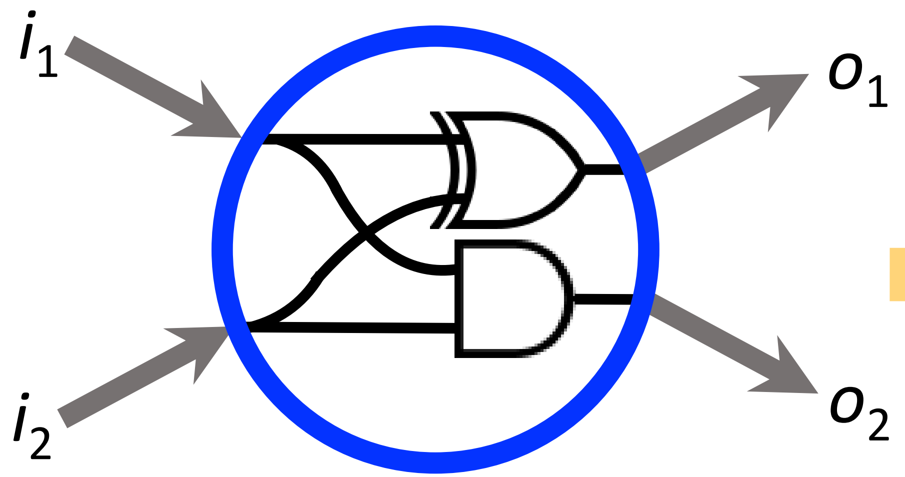
Bits:	Boolean circuits compute
➡ **Tiles:**	**Tile self-assembly implements circuits**
DNA:	DNA strands implement tiles

Gates → Tiles

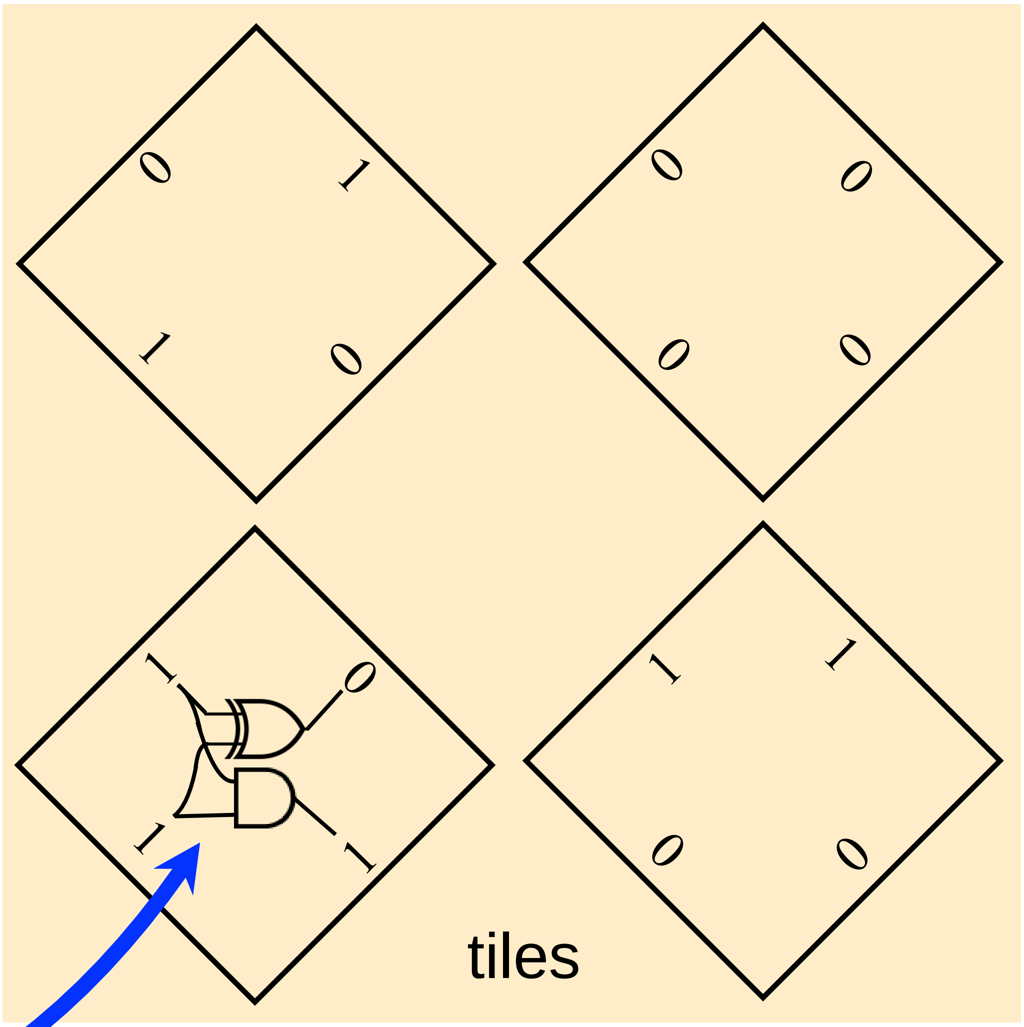


i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Gates → Tiles

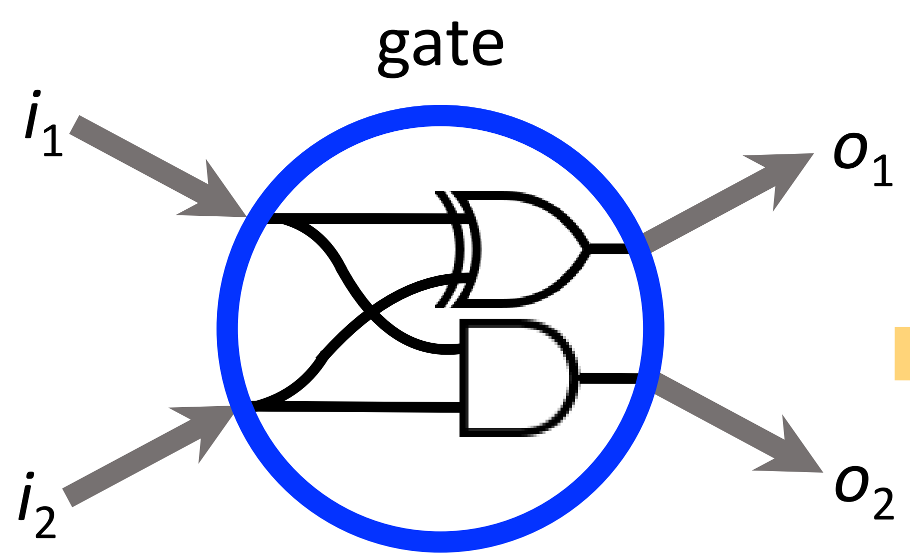


i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



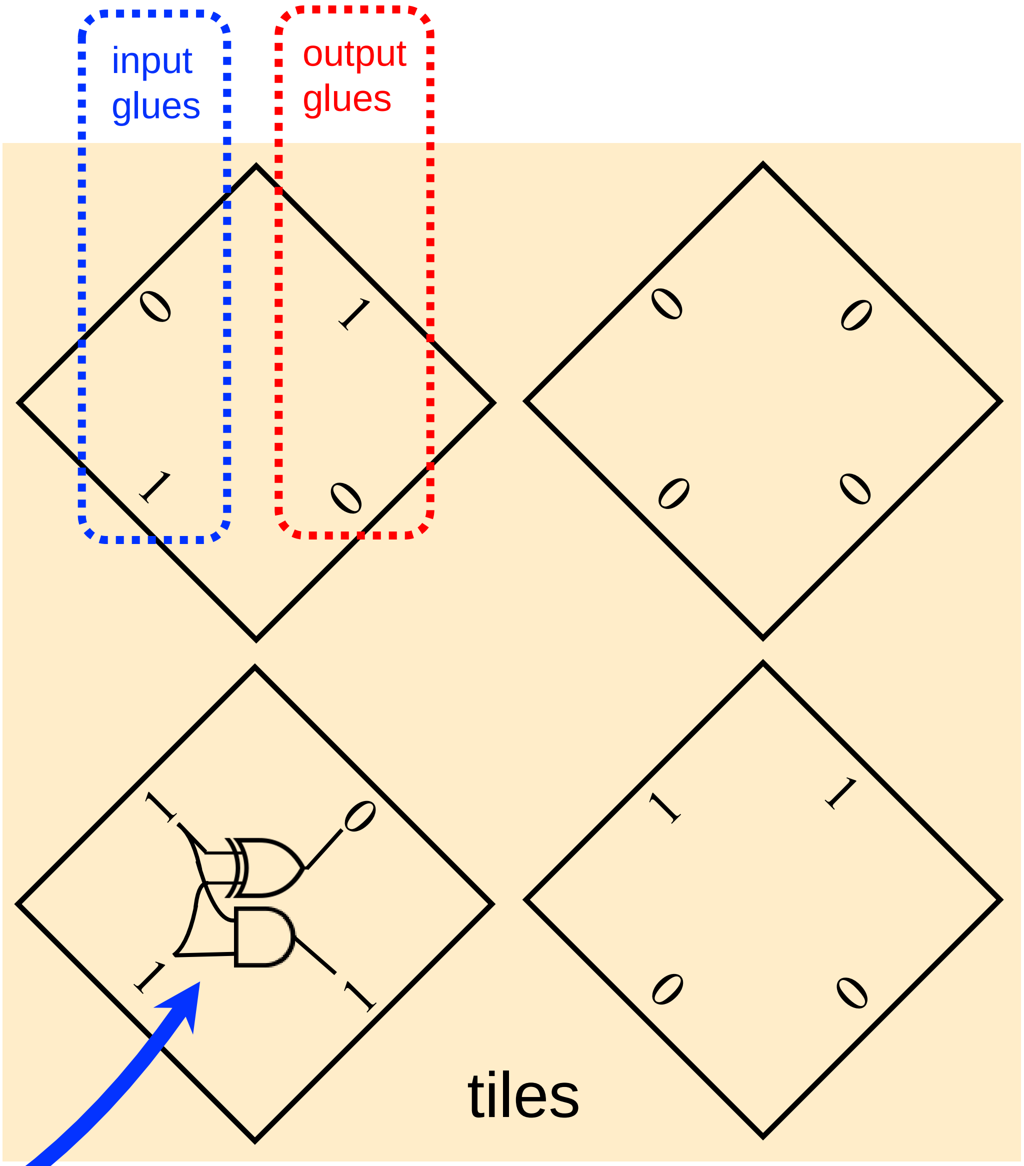
truth table row is encoded by a **tile** with 4 **glues** encoding bits

Gates → Tiles

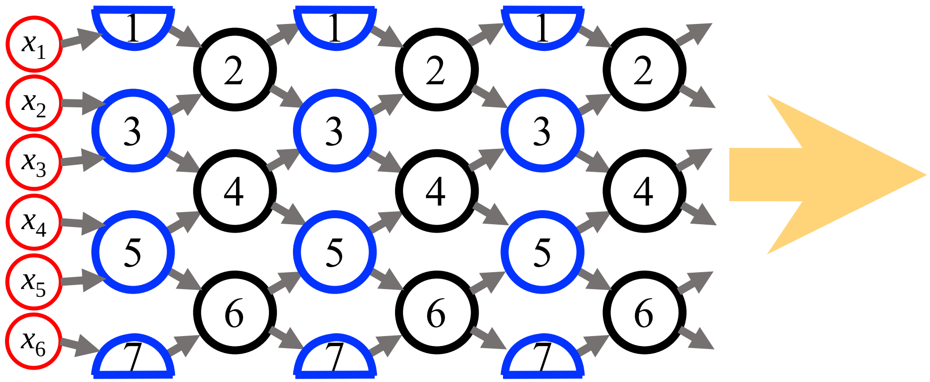


i_1	i_2	o_1	o_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

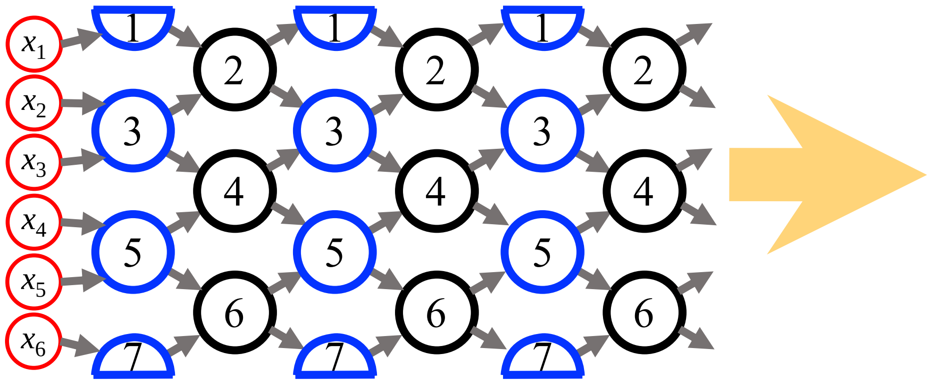
truth table row is encoded by a **tile** with 4 **glues** encoding bits



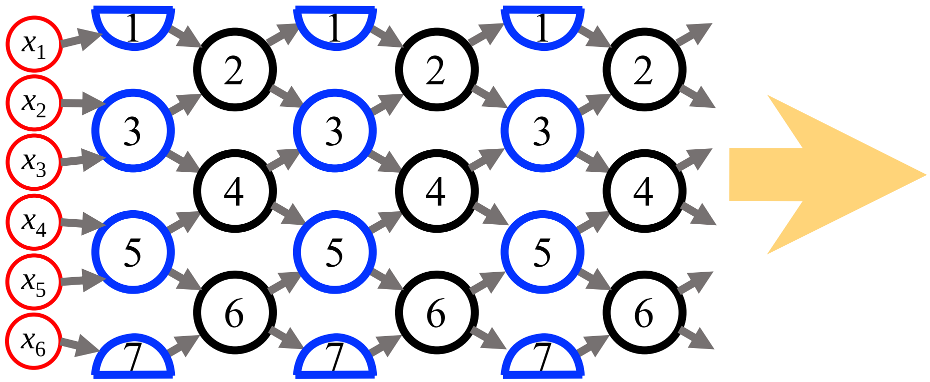
How tiles compute while growing (algorithmic self-assembly)

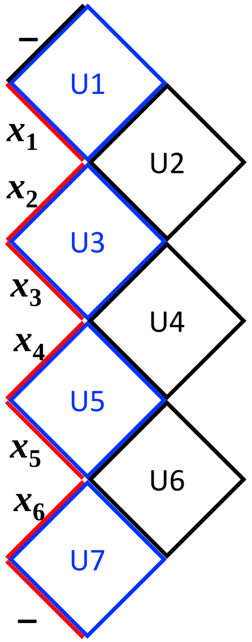


How tiles compute while growing (algorithmic self-assembly)

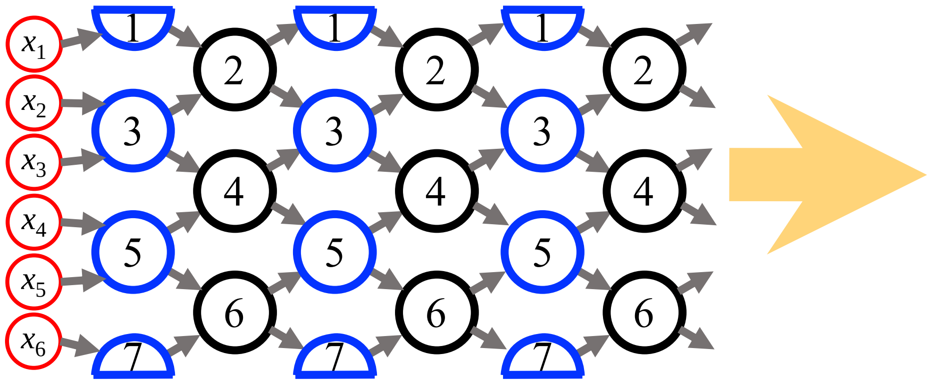


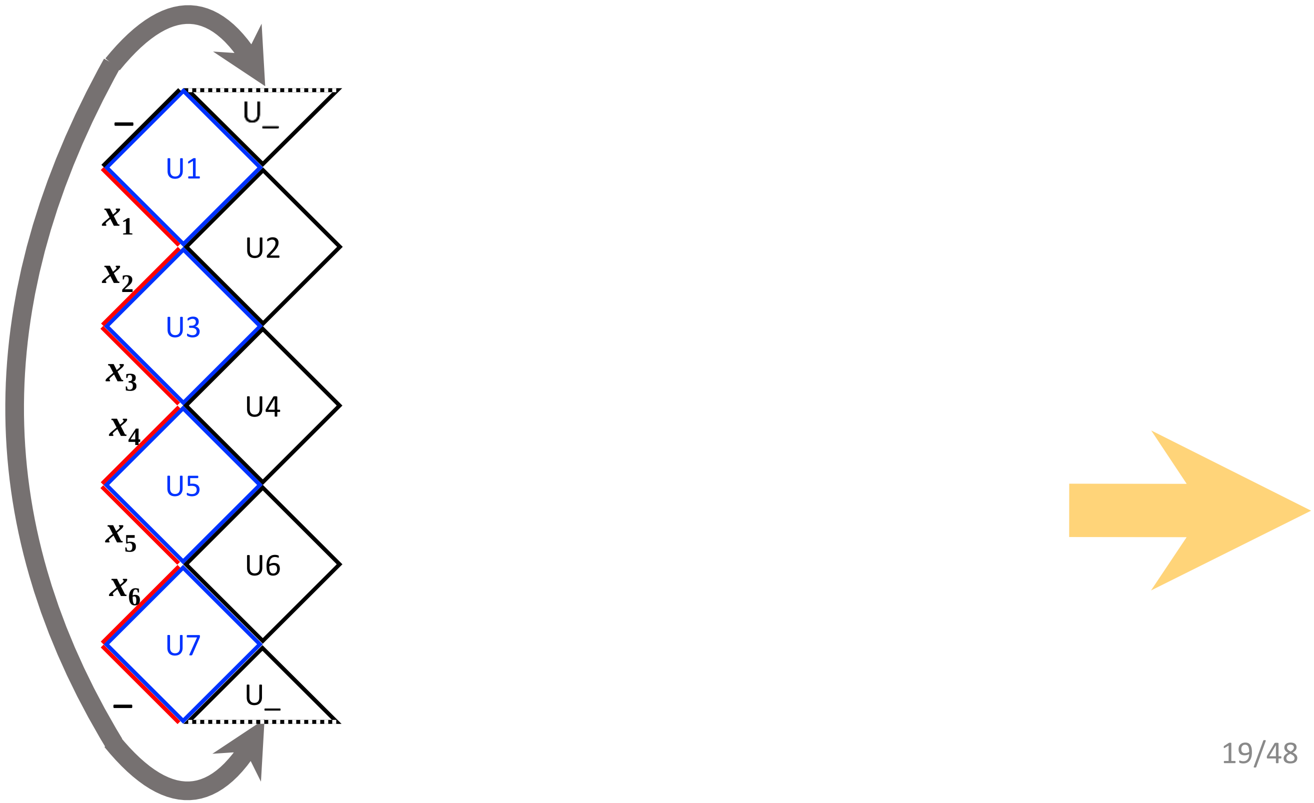
How tiles compute while growing (algorithmic self-assembly)



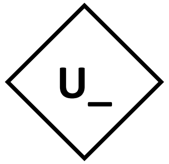


How tiles compute while growing (algorithmic self-assembly)

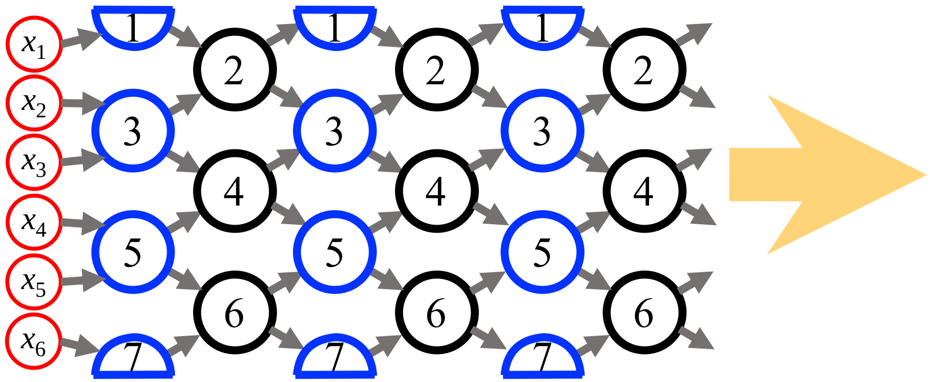


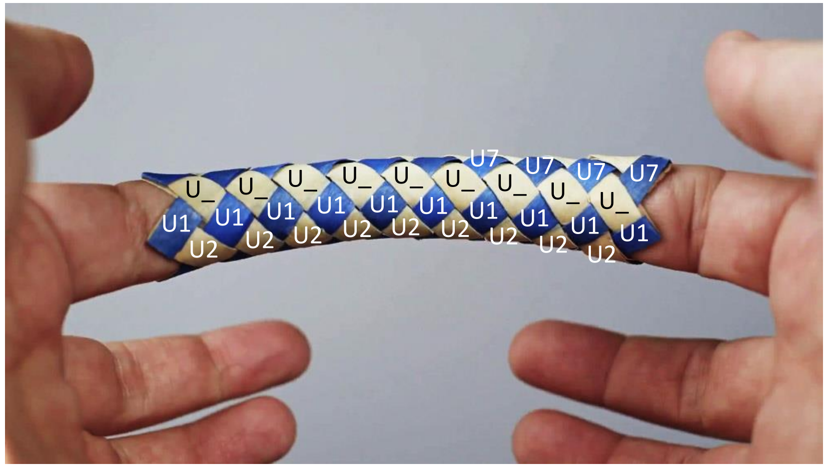


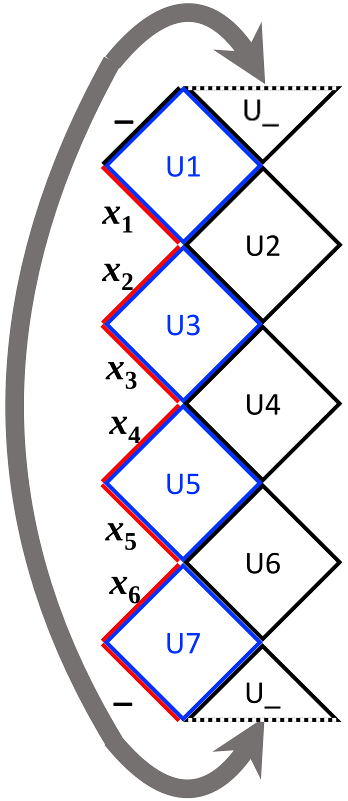
“data-free” tile wraps top to bottom to form a tube



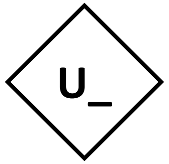
How tiles compute while growing (algorithmic self-assembly)



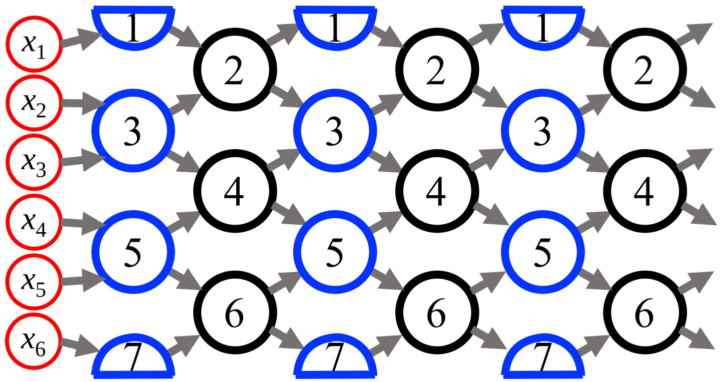


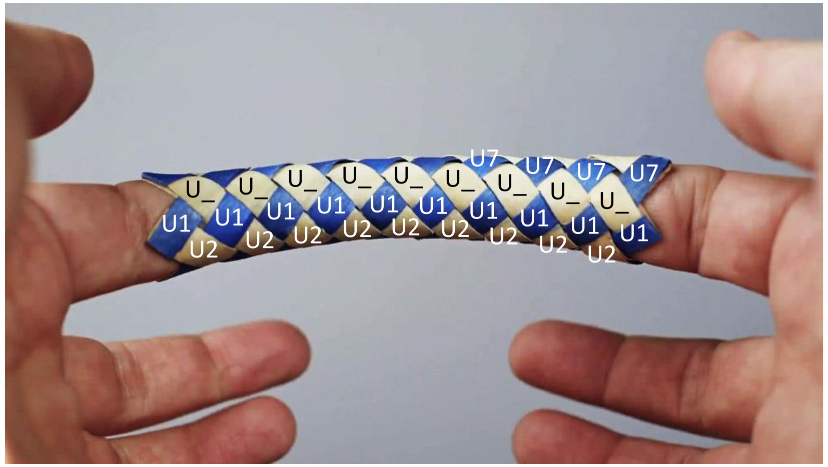


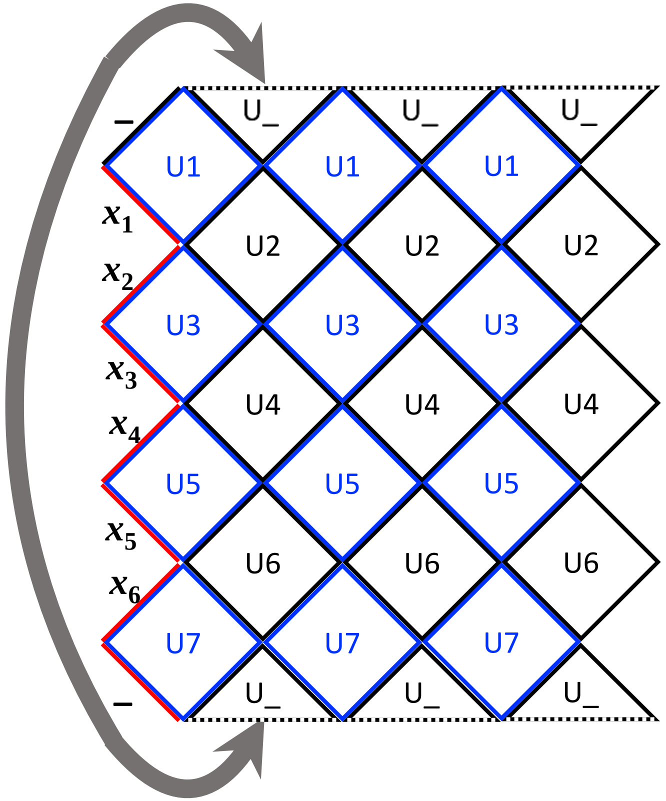
"data-free" tile wraps top to bottom to form a tube



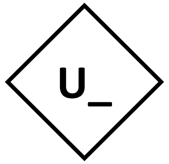
How tiles compute while growing (algorithmic self-assembly)



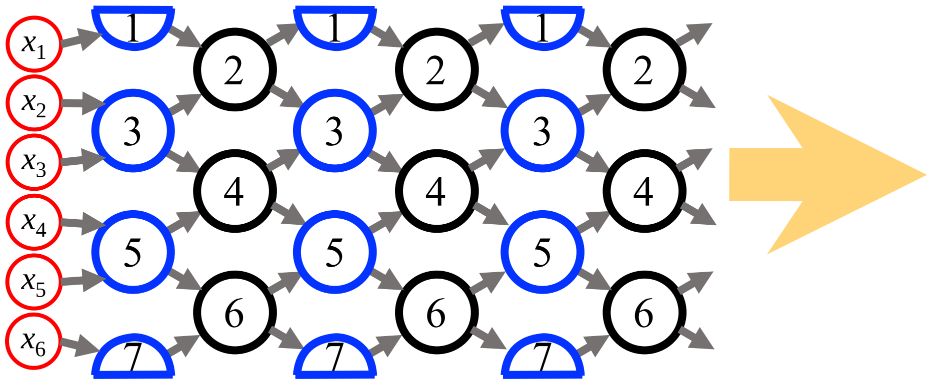


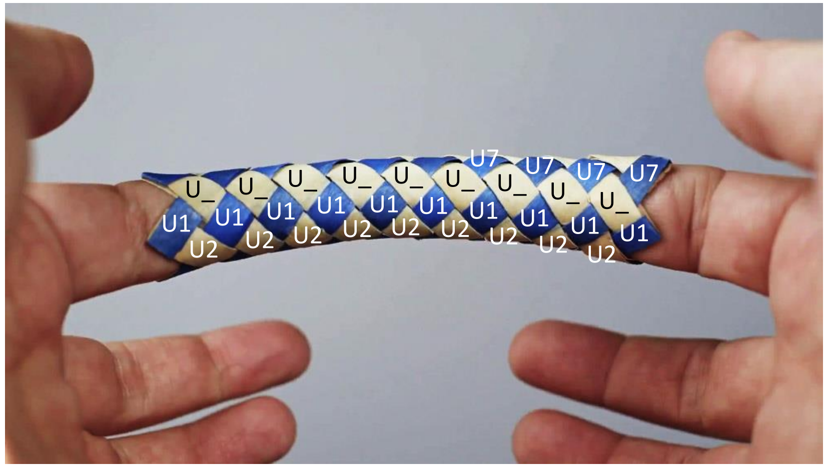


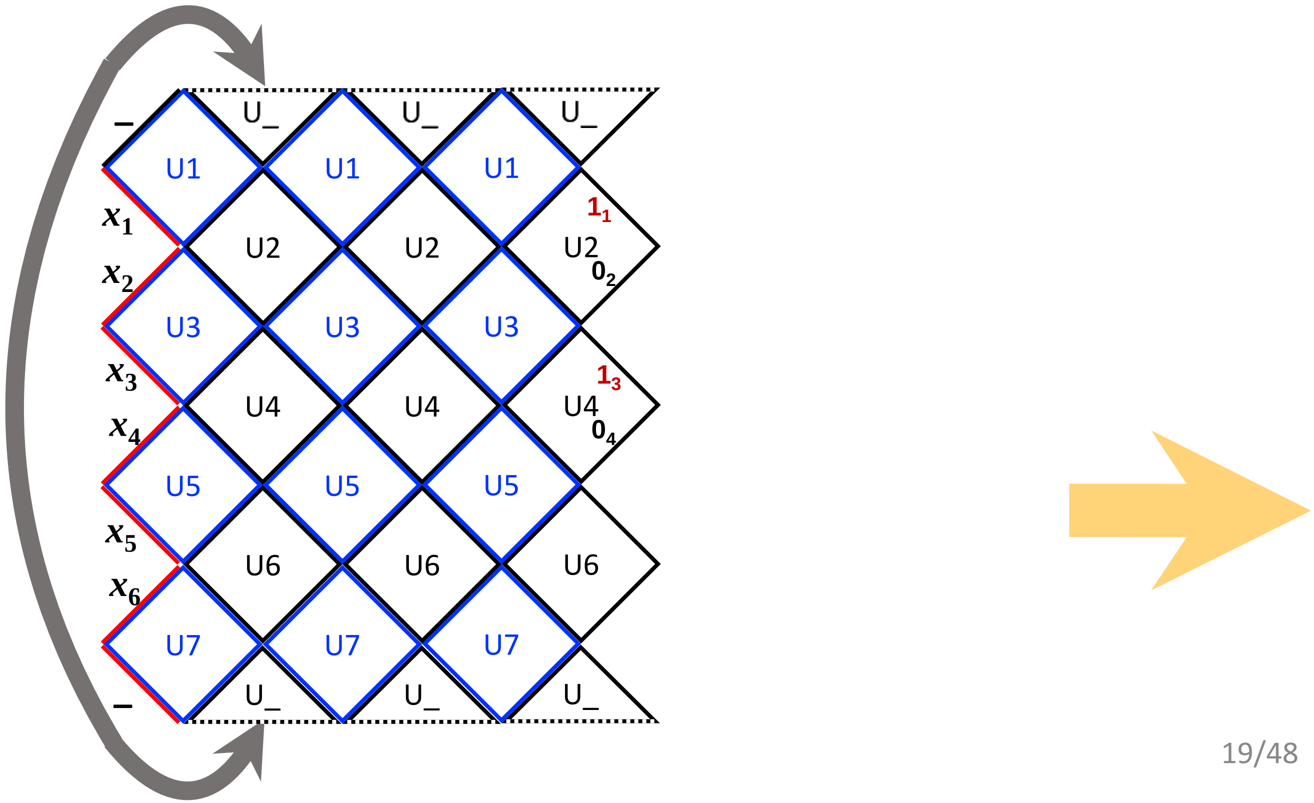
“data-free” tile wraps top to bottom to form a tube



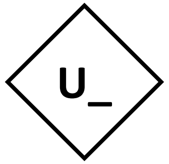
How tiles compute while growing (algorithmic self-assembly)



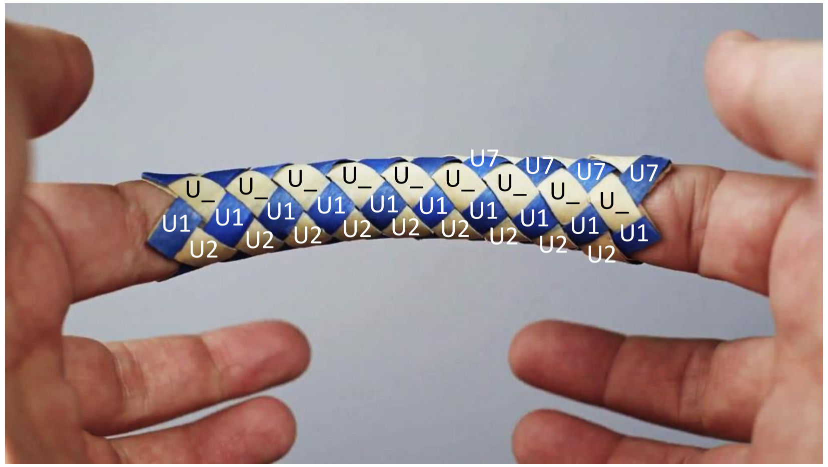


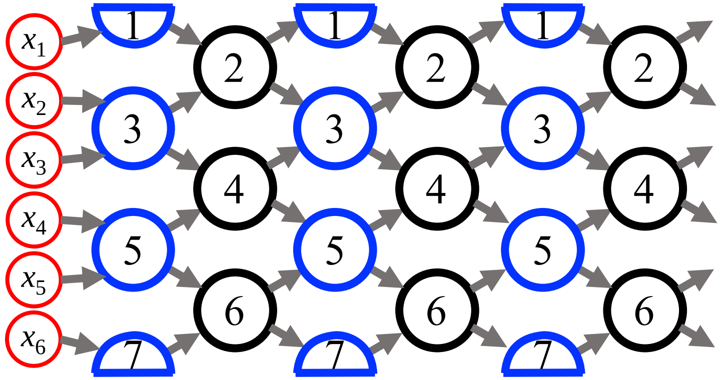


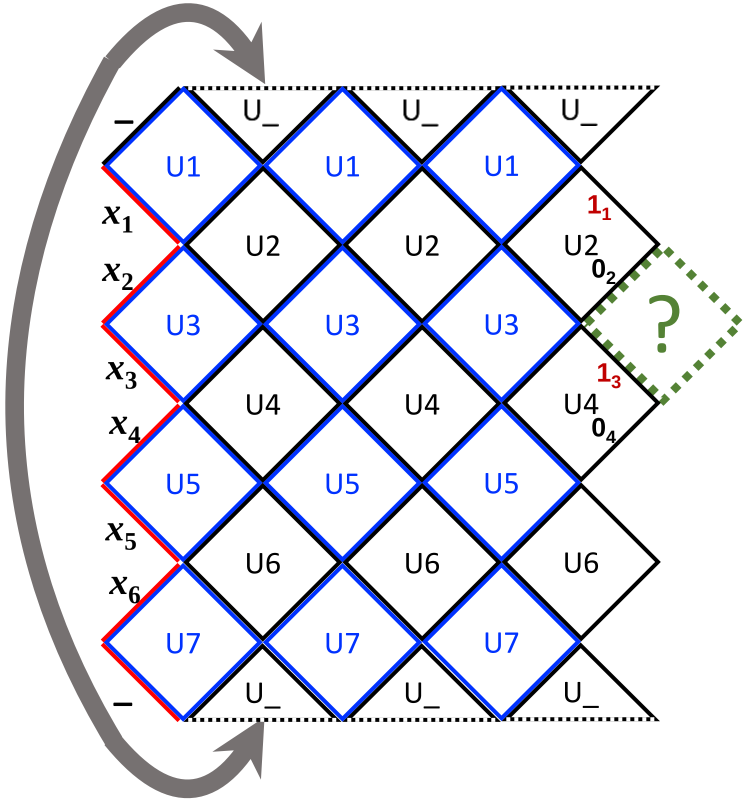
“data-free” tile wraps top to bottom to form a tube



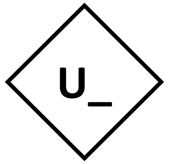
How tiles compute while growing (algorithmic self-assembly)



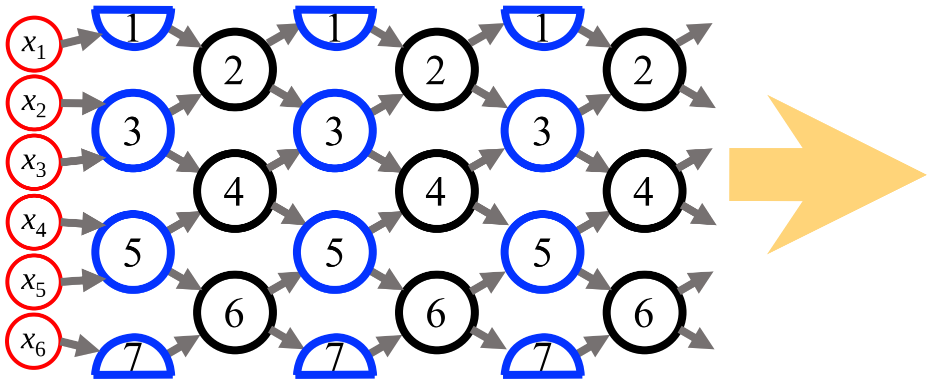


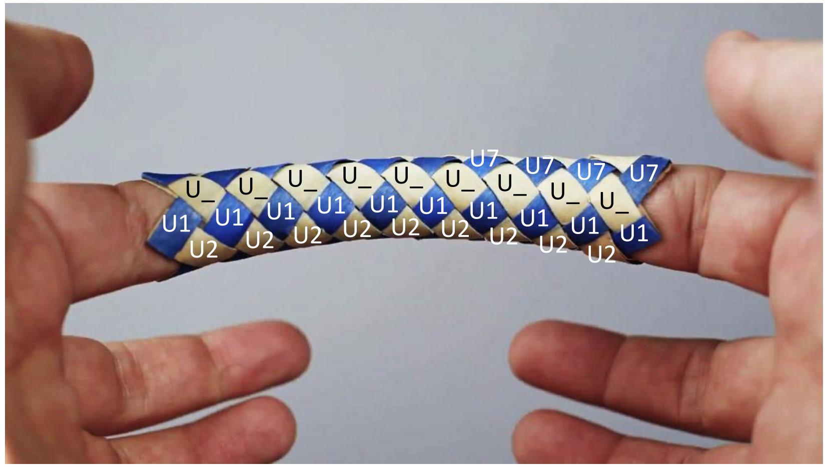


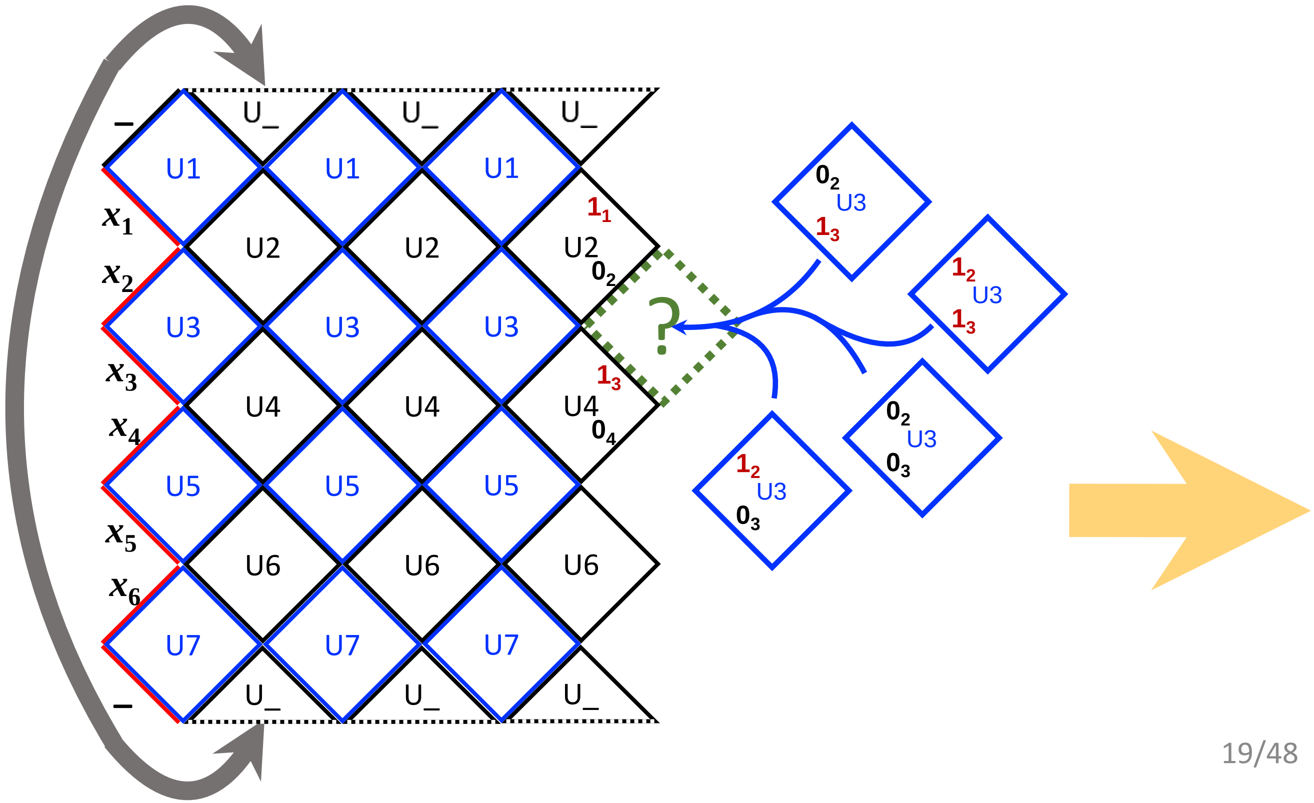
"data-free" tile wraps top to bottom to form a tube



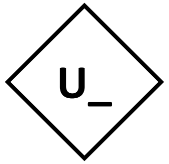
How tiles compute while growing (algorithmic self-assembly)



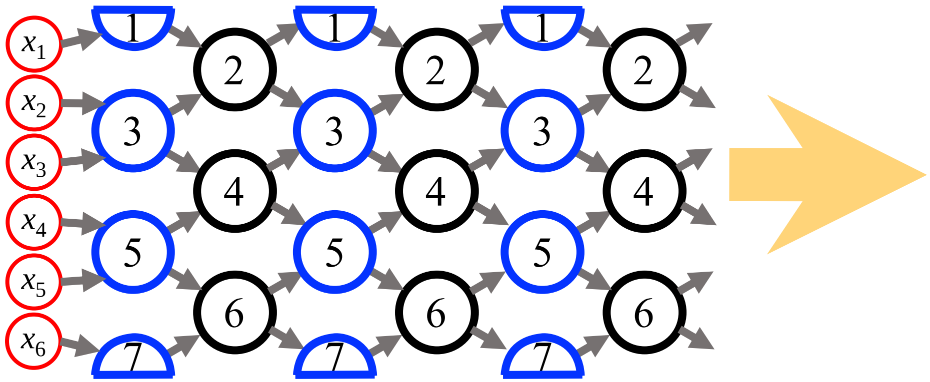


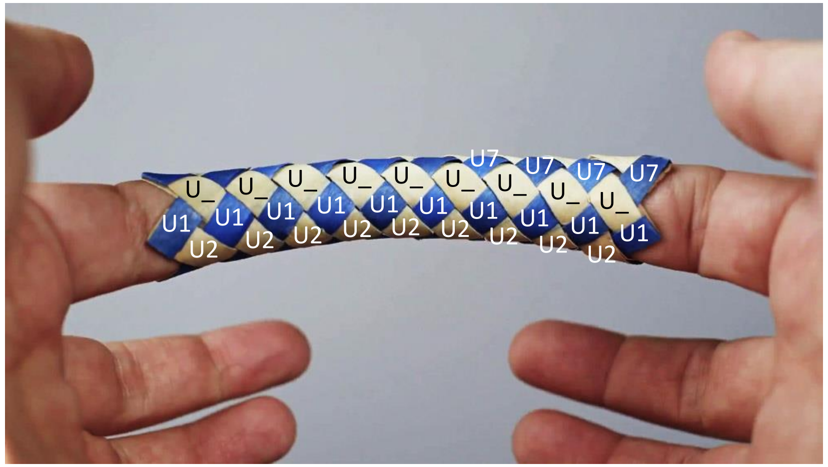


“data-free” tile wraps top to bottom to form a tube

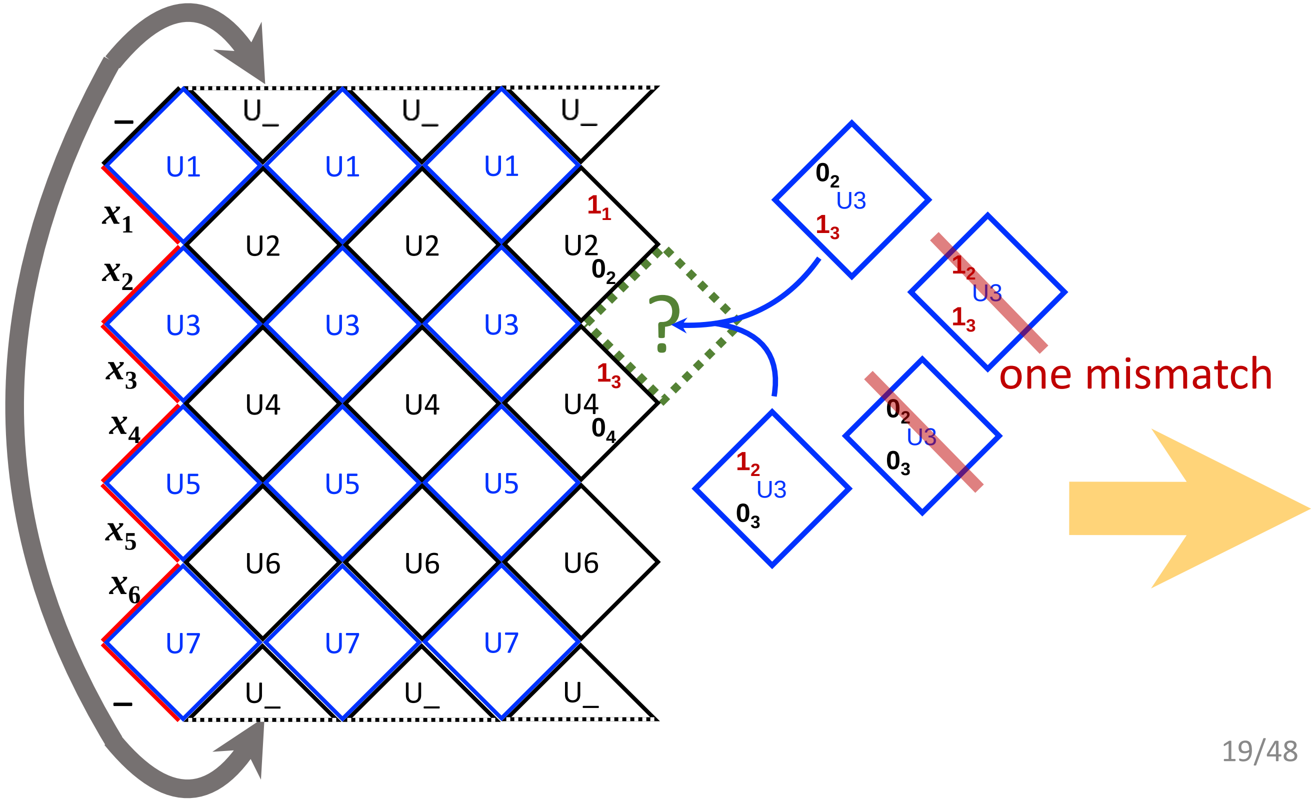


How tiles compute while growing (algorithmic self-assembly)

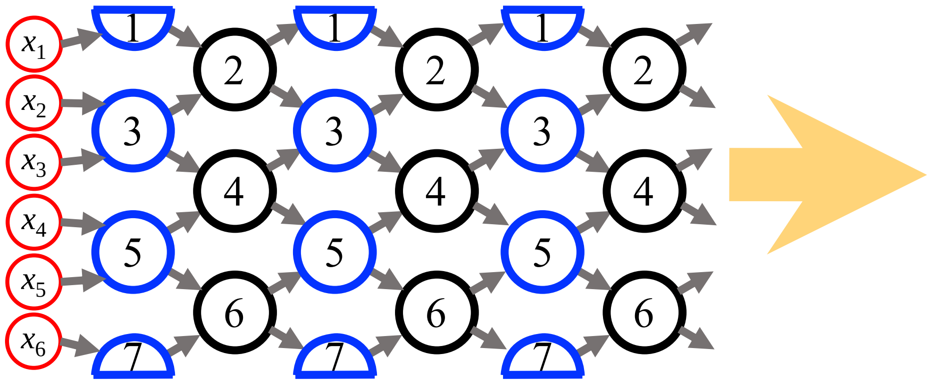


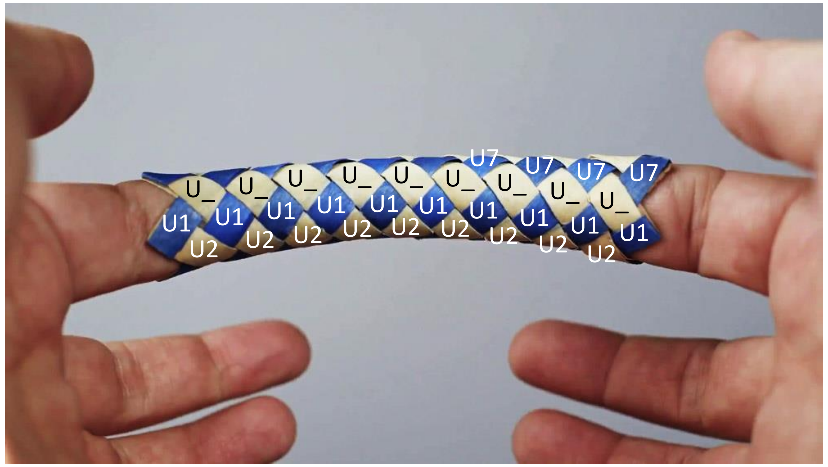


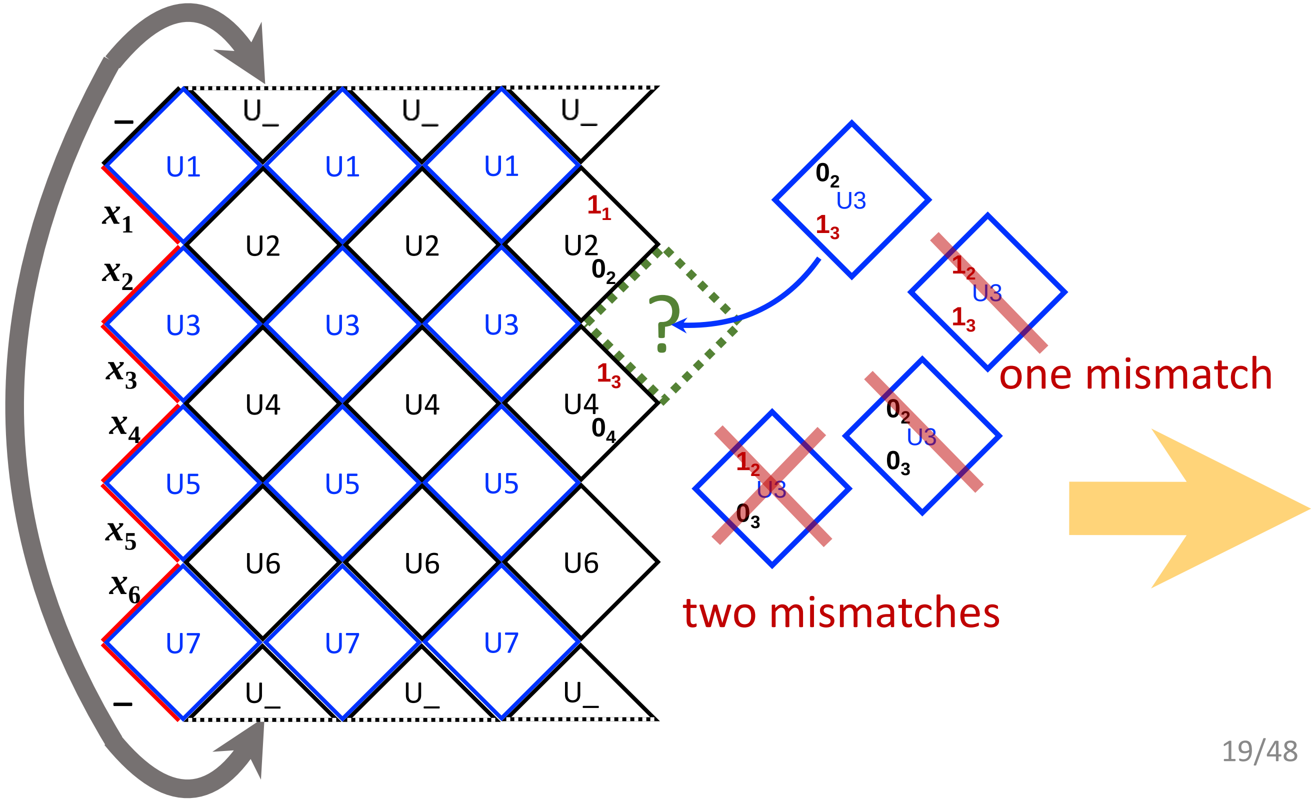
“data-free” tile wraps top to bottom to form a tube



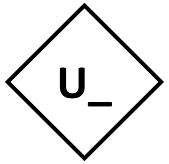
How tiles compute while growing (algorithmic self-assembly)



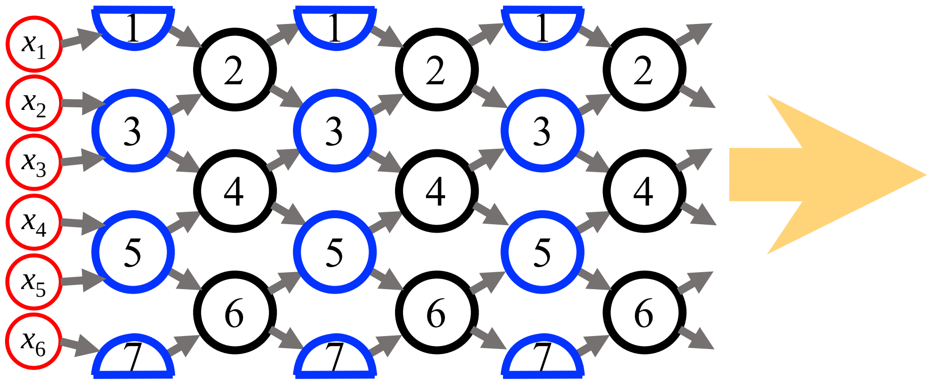


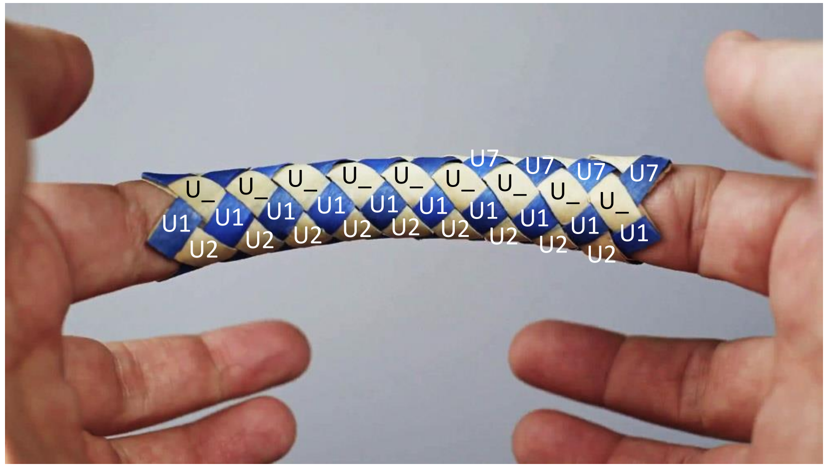


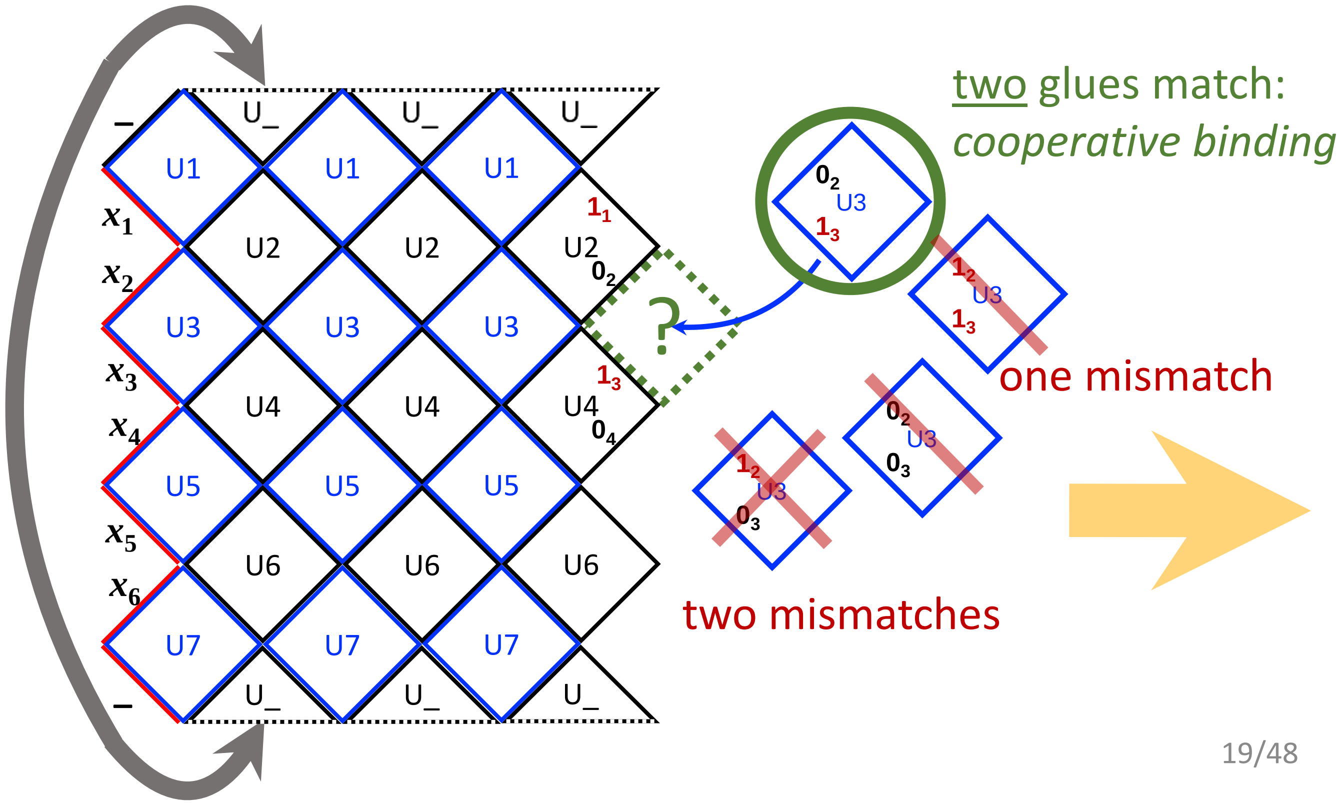
"data-free" tile wraps top to bottom to form a tube



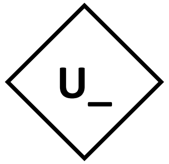
How tiles compute while growing (algorithmic self-assembly)



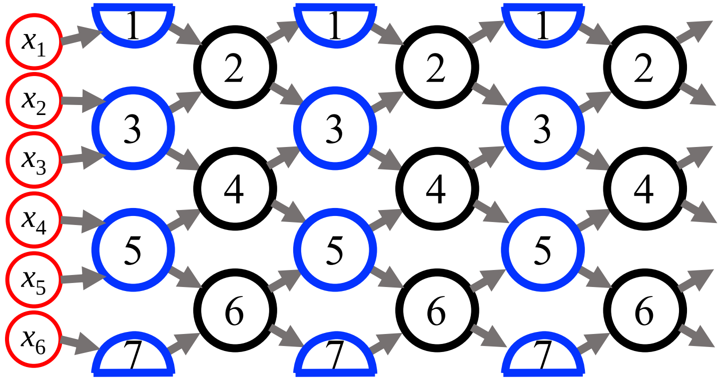


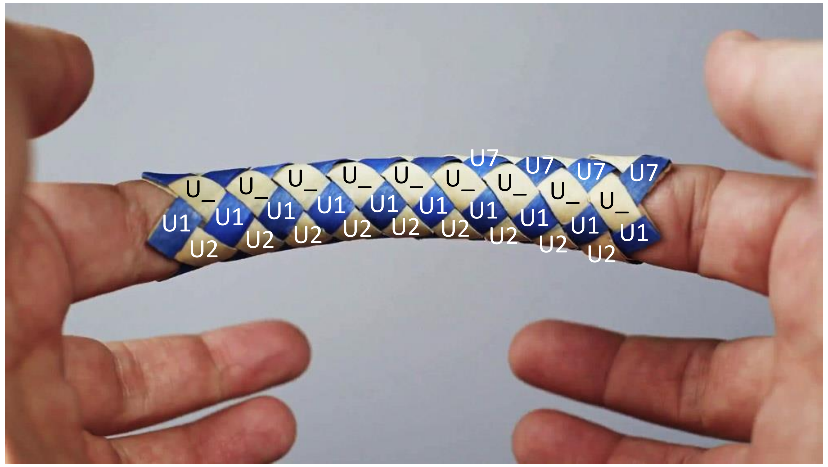


"data-free" tile wraps top to bottom to form a tube

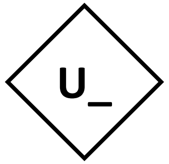


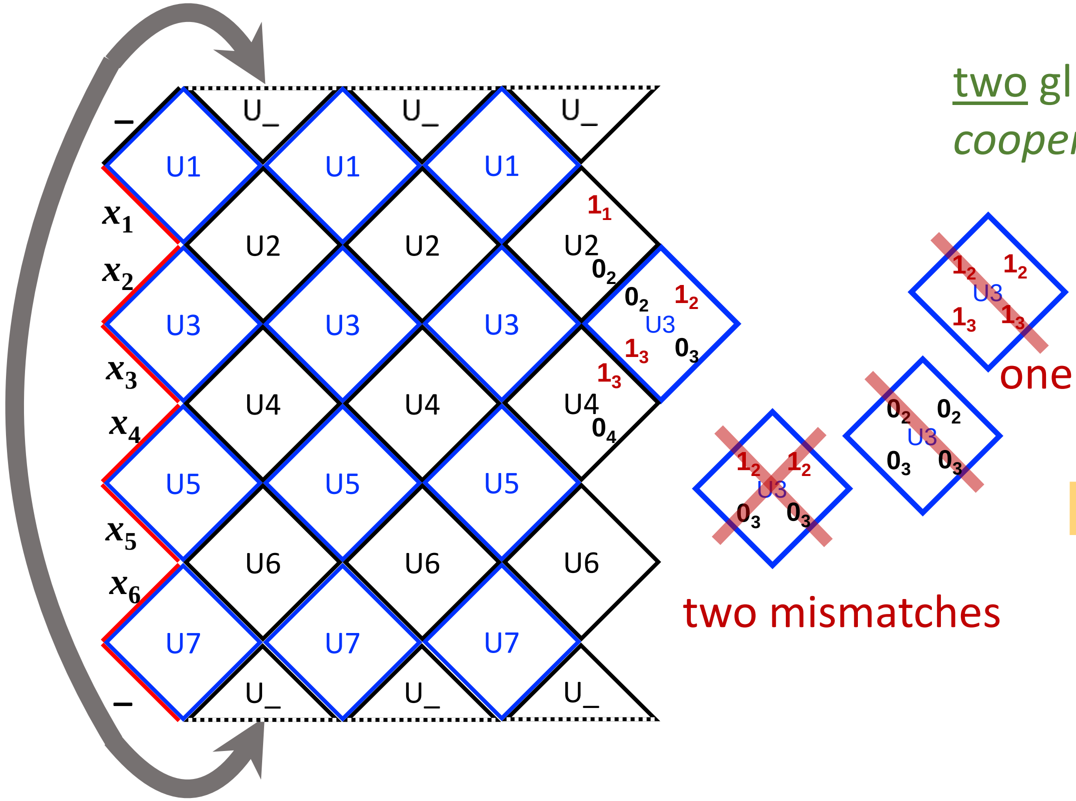
How tiles compute while growing (algorithmic self-assembly)





“data-free” tile wraps top to bottom to form a tube





two glues match: *cooperative binding*

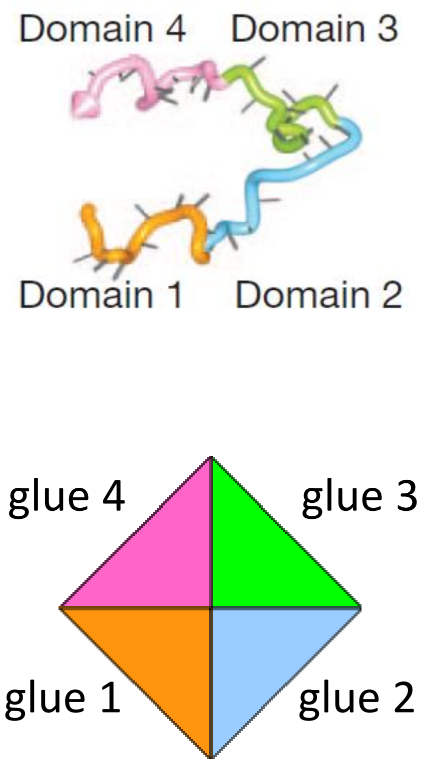
one mismatch

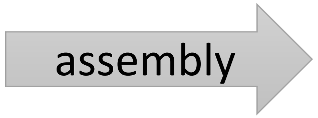
two mismatches

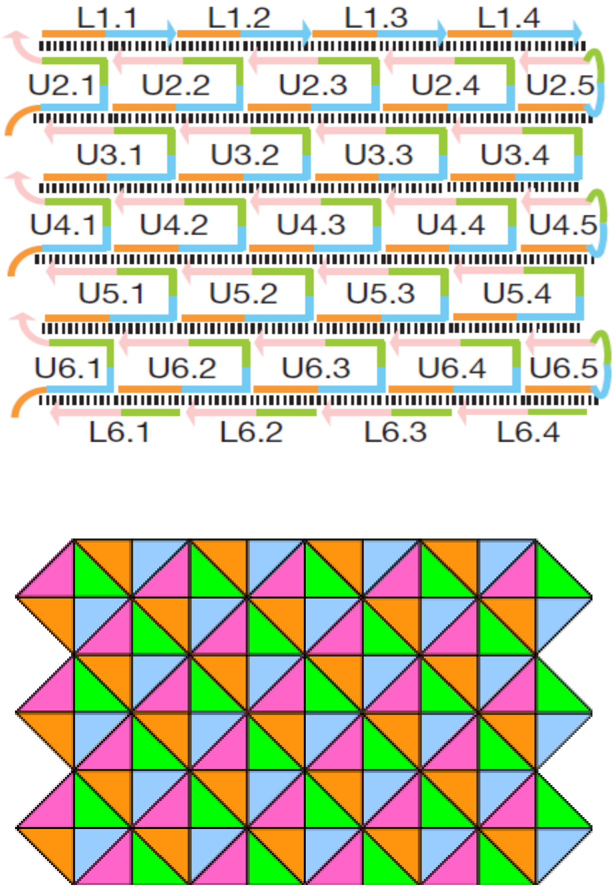
Hierarchy of abstractions

Bits:	Boolean circuits compute
Tiles:	Tile self-assembly implements circuits
➡ **DNA:**	**DNA strands implement tiles**

DNA single-stranded tiles

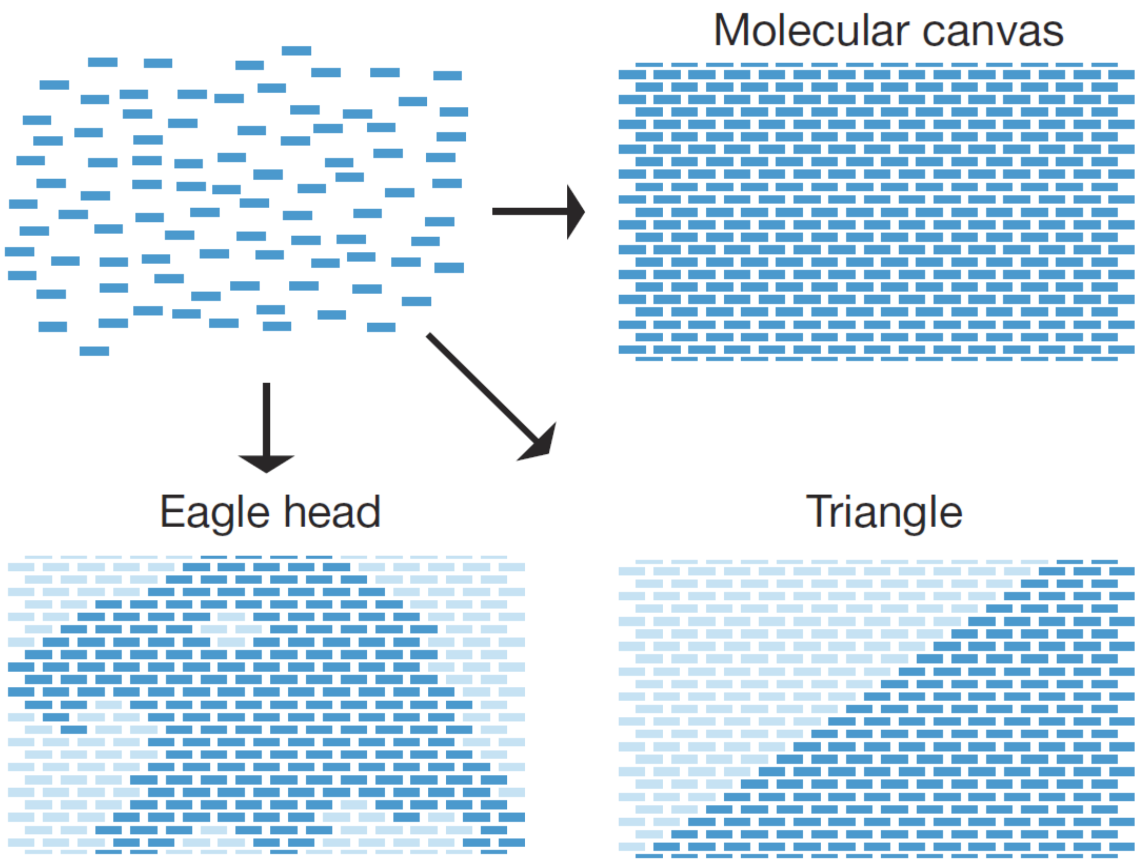






Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif
Programming DNA tube circumferences
Science 321, 824-826 (2008)

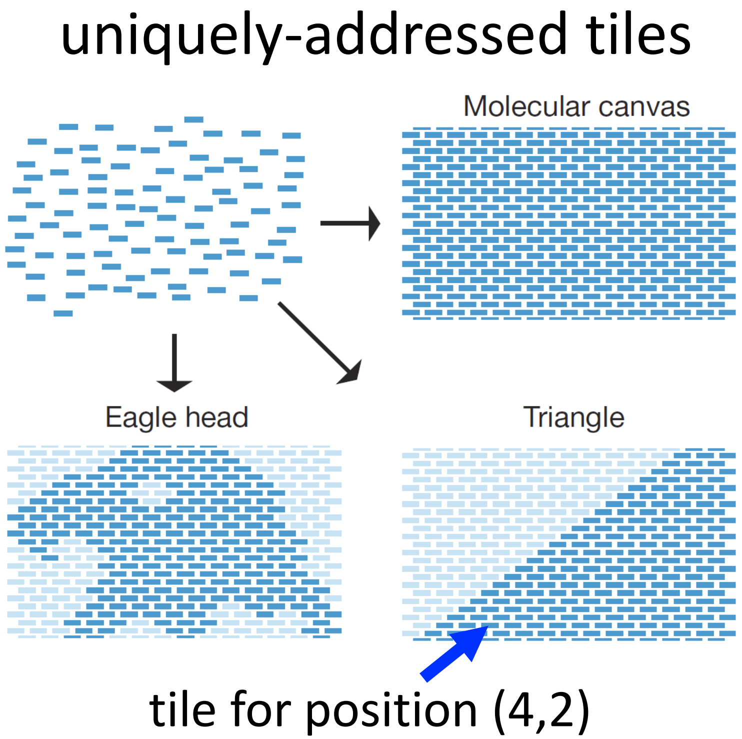
Single-stranded tiles for making any shape

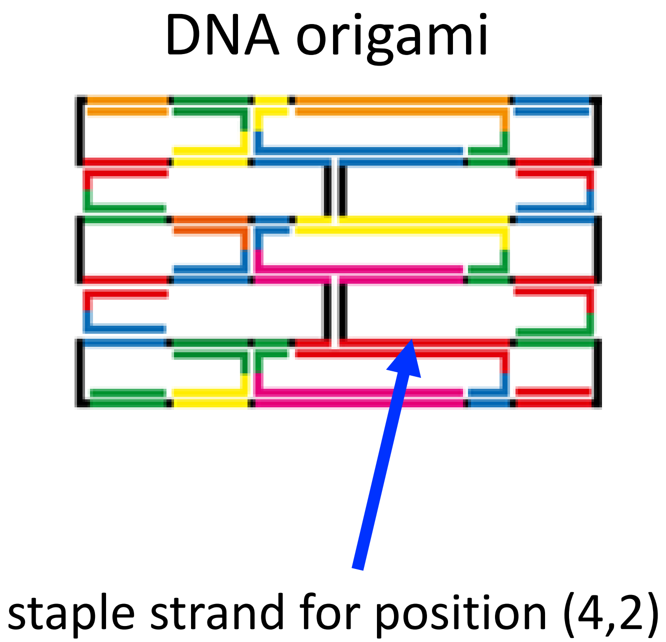


Bryan Wei, Mingjie Dai, and Peng Yin. *Complex shapes self-assembled from single-stranded DNA tiles*. Nature 2012.

Uniquely addressed self-assembly versus algorithmic

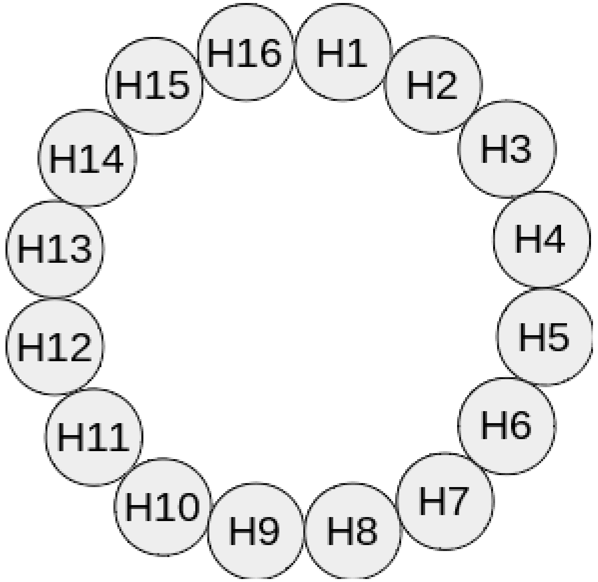
Unique addressing: each DNA "monomer" appears **exactly once** in final structure:



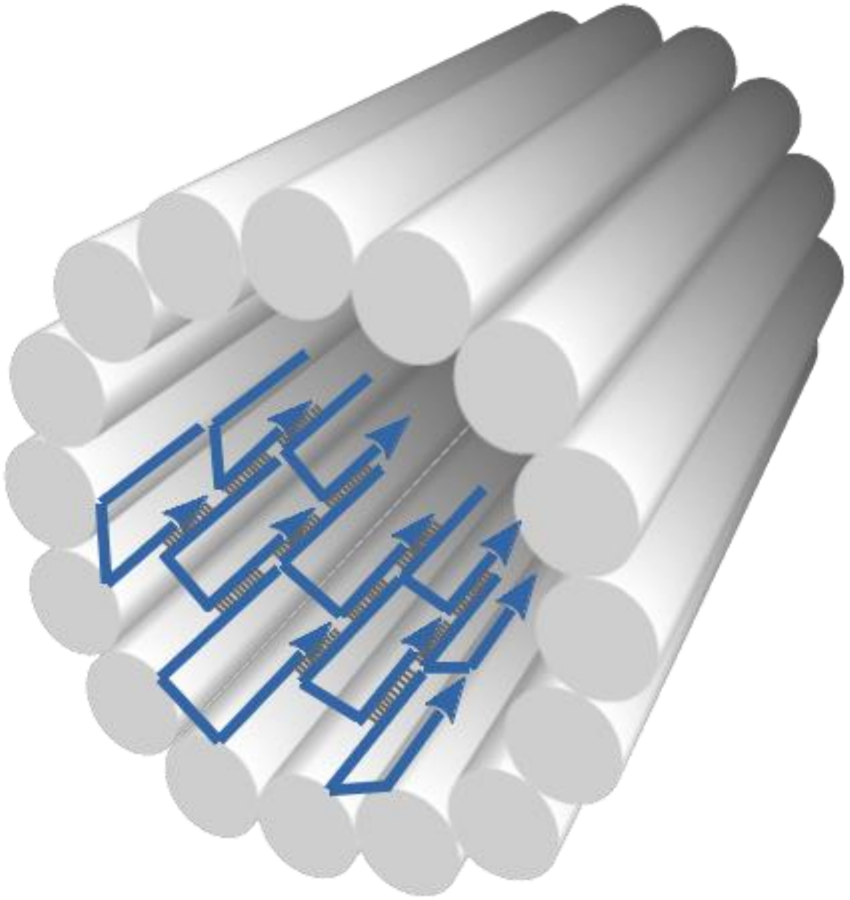


Algorithmic: DNA tiles are **reused** throughout the structure.

Single-stranded tile tubes

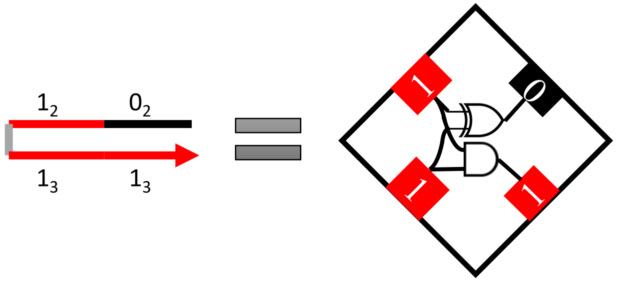


DNA-level diagram of 20-helix tube

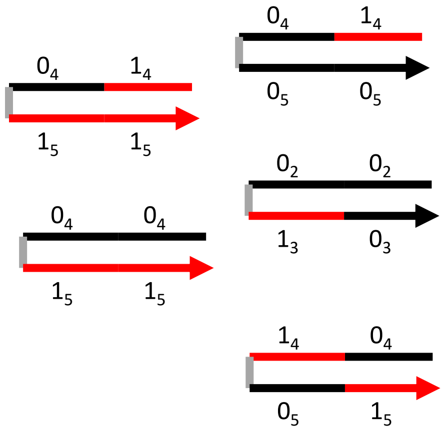


Yin, Hariadi, Sahu, Choi, Park, LaBean, Reif, Programming DNA tube circumferences, *Science* 321, 824-826 (2008)

Seeded growth

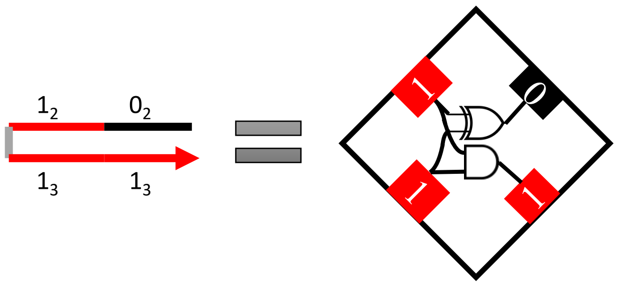


single-stranded tiles
implementing circuit gates

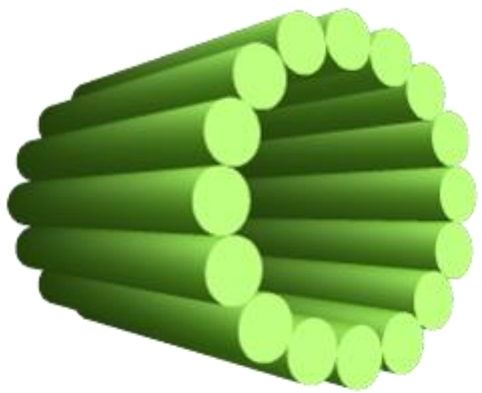


need barrier to <u>nucleation</u>
(tile growth without seed);
[tile]=100 nM;
temperature=50.9° C

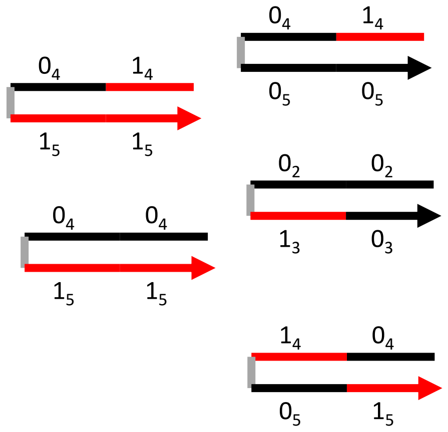
Seeded growth



DNA origami seed

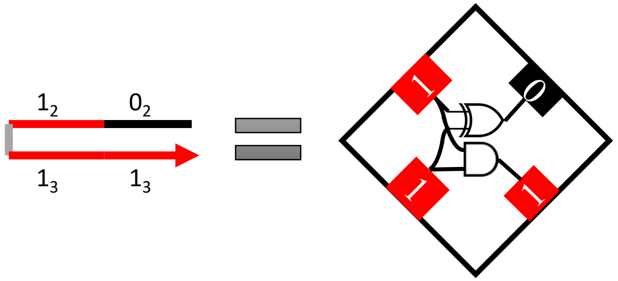


single-stranded tiles
implementing circuit gates



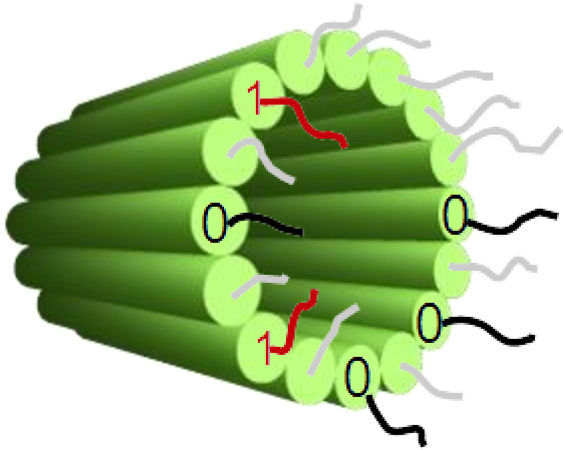
need barrier to <u>nucleation</u>
(tile growth without seed);
[tile]=100 nM;
temperature=50.9° C

Seeded growth

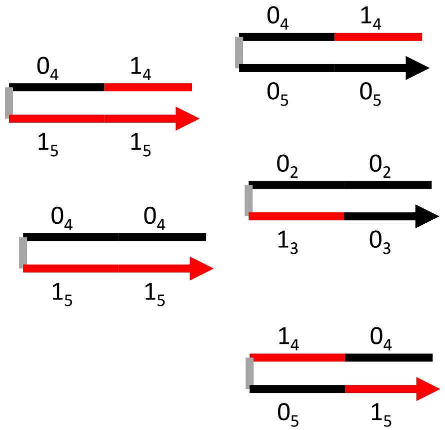


DNA origami seed

single-stranded “input-adapter” extensions encoding 6 input bits

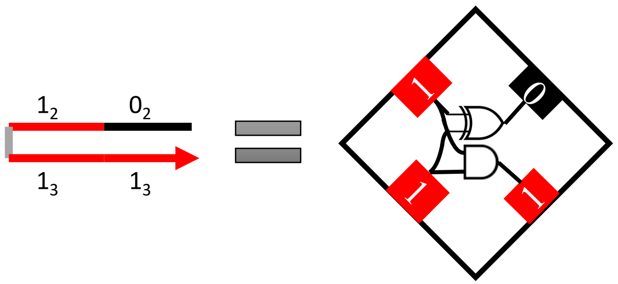


single-stranded tiles implementing circuit gates



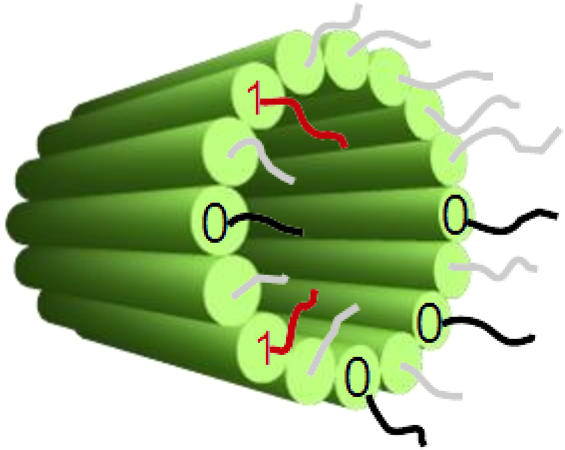
need barrier to <u>nucleation</u>
(tile growth without seed);
[tile]=100 nM;
temperature=50.9° C

Seeded growth

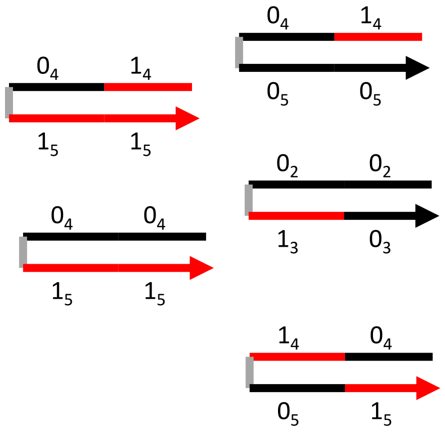


DNA origami seed

single-stranded “input-adapter” extensions encoding 6 input bits

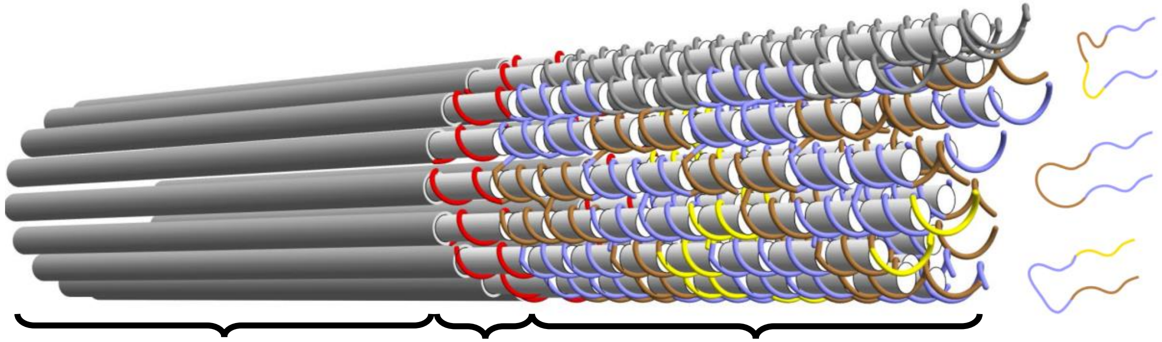


single-stranded tiles implementing circuit gates

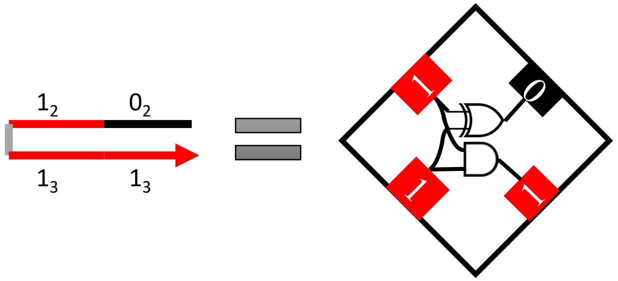


need barrier to nucleation (tile growth without seed);
[tile]=100 nM;
temperature=50.9° C

hold 8-48 hours

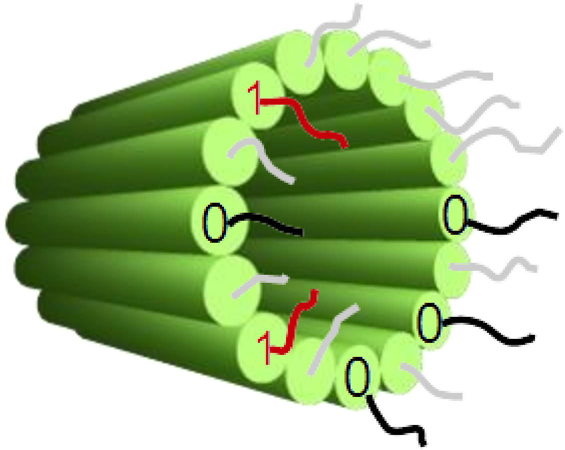


Seeded growth

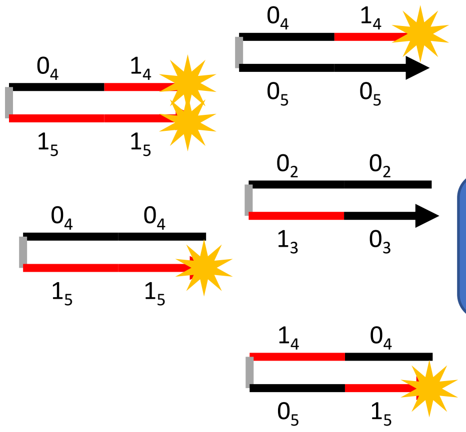


DNA origami seed

single-stranded “input-adapter” extensions encoding 6 input bits



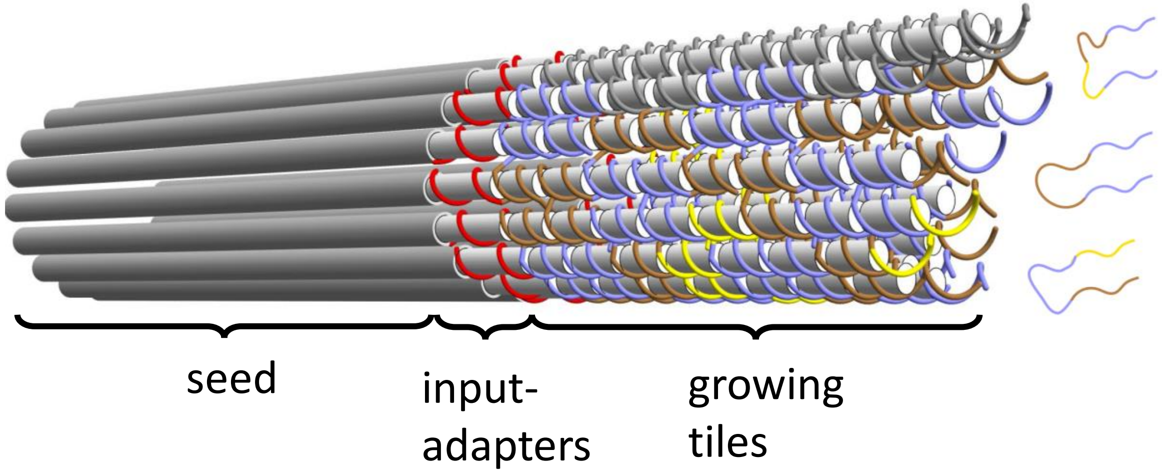
single-stranded tiles implementing circuit gates



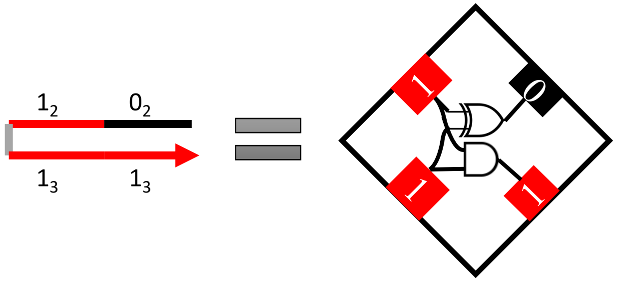
biotins where output = 1

need barrier to nucleation (tile growth without seed); [tile]=100 nM; temperature=50.9° C

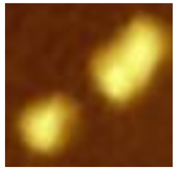
hold 8-48 hours



Seeded growth

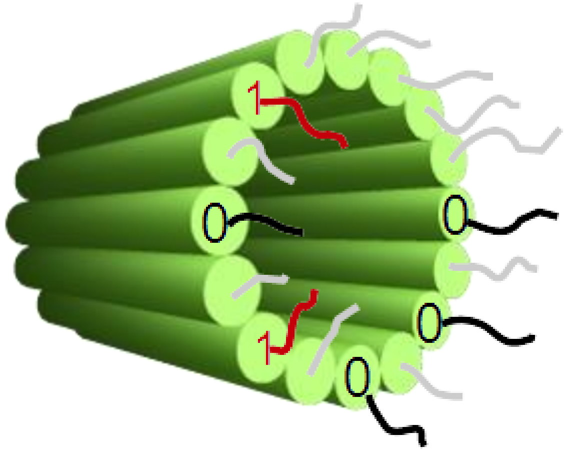


can later add streptavidin (5 nm wide protein) to bind biotins and visualize where the 1's are

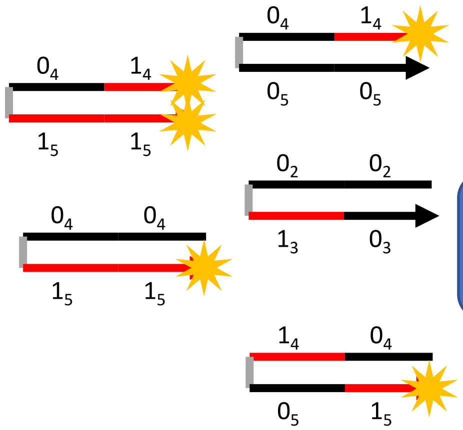


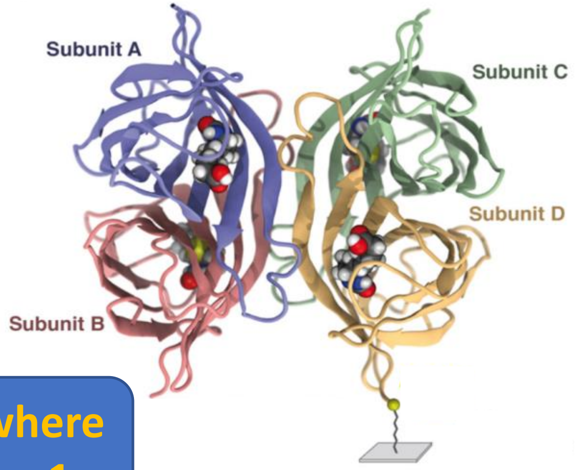
DNA origami seed

single-stranded "input-adapter" extensions encoding 6 input bits



single-stranded tiles implementing circuit gates

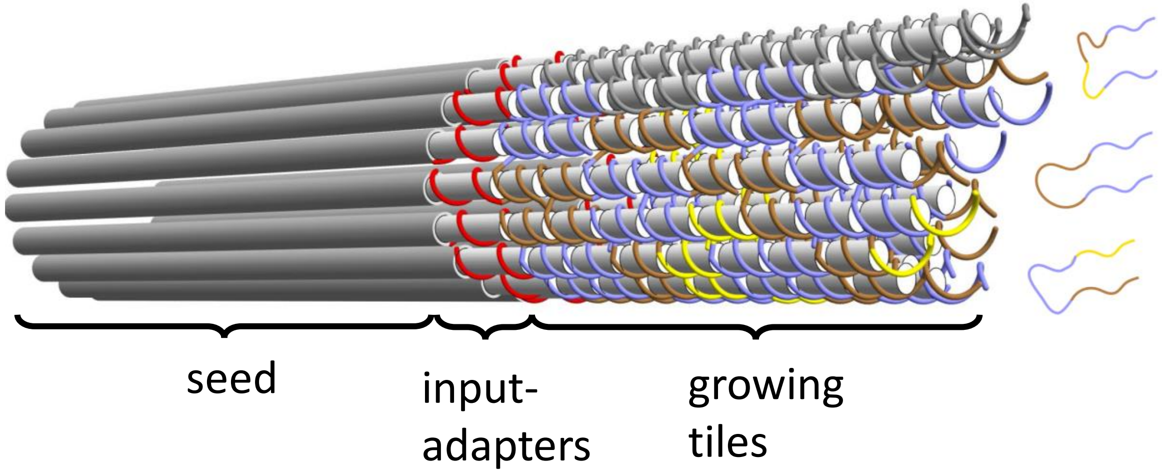




biotins where output = 1

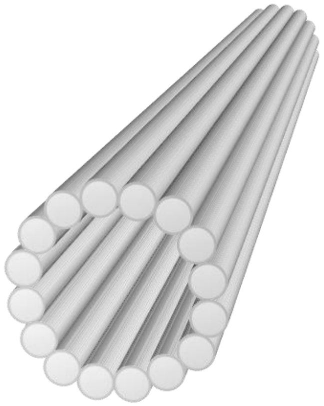
need barrier to <u>nucleation</u> (tile growth without seed);
[tile]=100 nM;
temperature=50.9° C

hold 8-48 hours

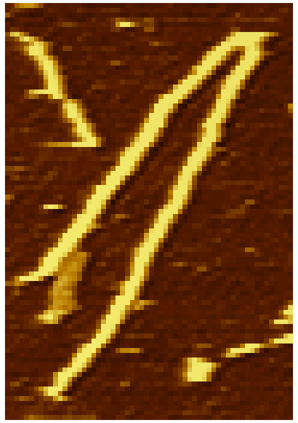


Tubes and ribbons

tube

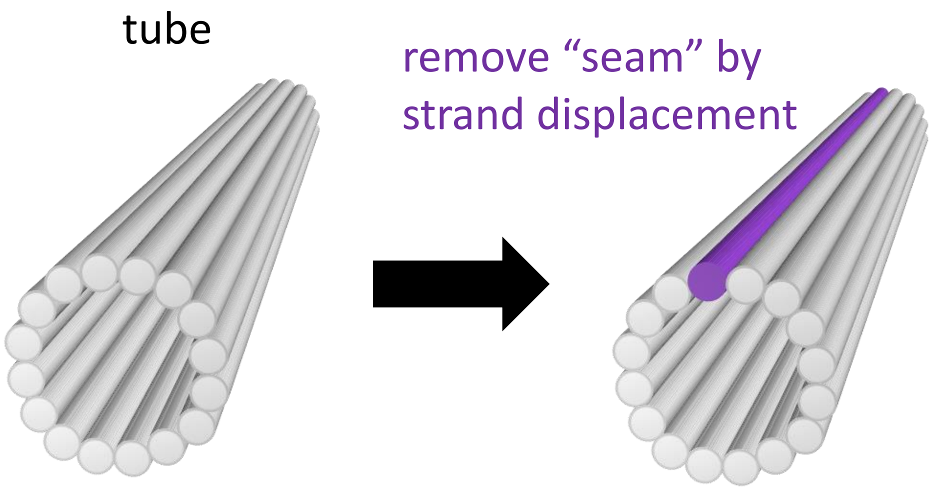


AFM image

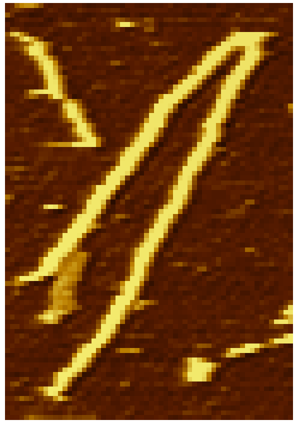


500 nm

Tubes and ribbons



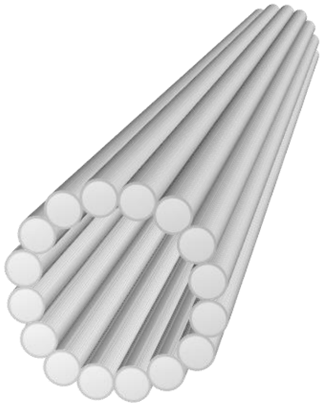
AFM
image



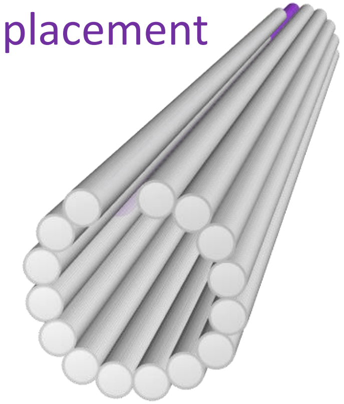
500 nm

Tubes and ribbons

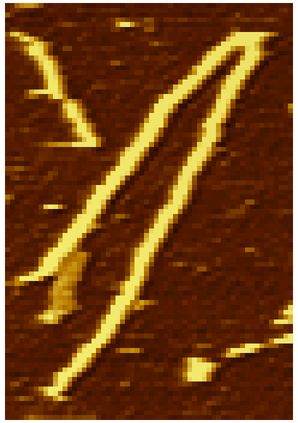
tube



remove “seam” by
strand displacement

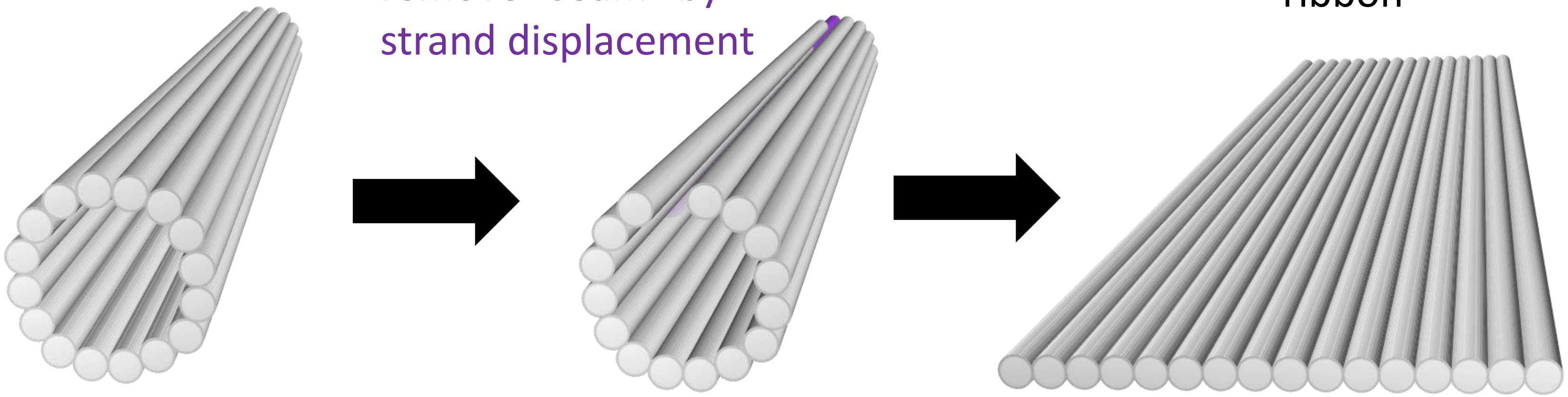


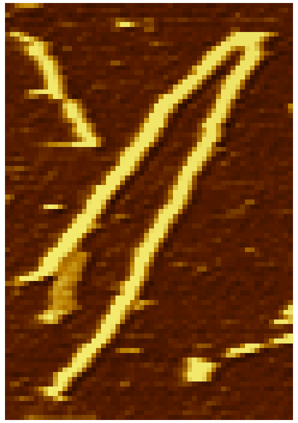
AFM
image

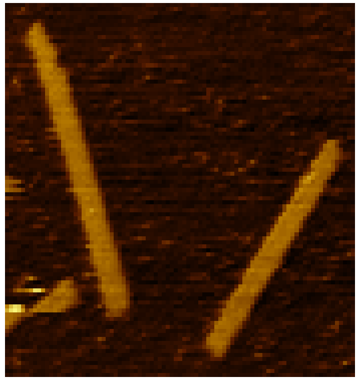


500 nm

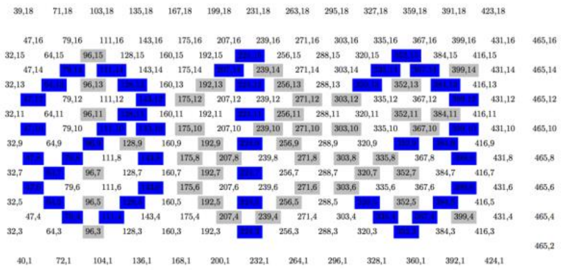
Tubes and ribbons





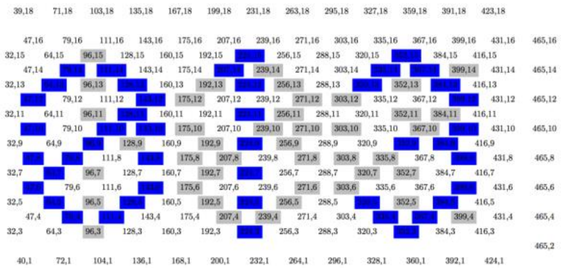


Bar-coding origami seed for imaging multiple samples at once



some staples of origami seed
have version with a biotin

Bar-coding origami seed for imaging multiple samples at once



some staples of origami seed
have version with a biotin

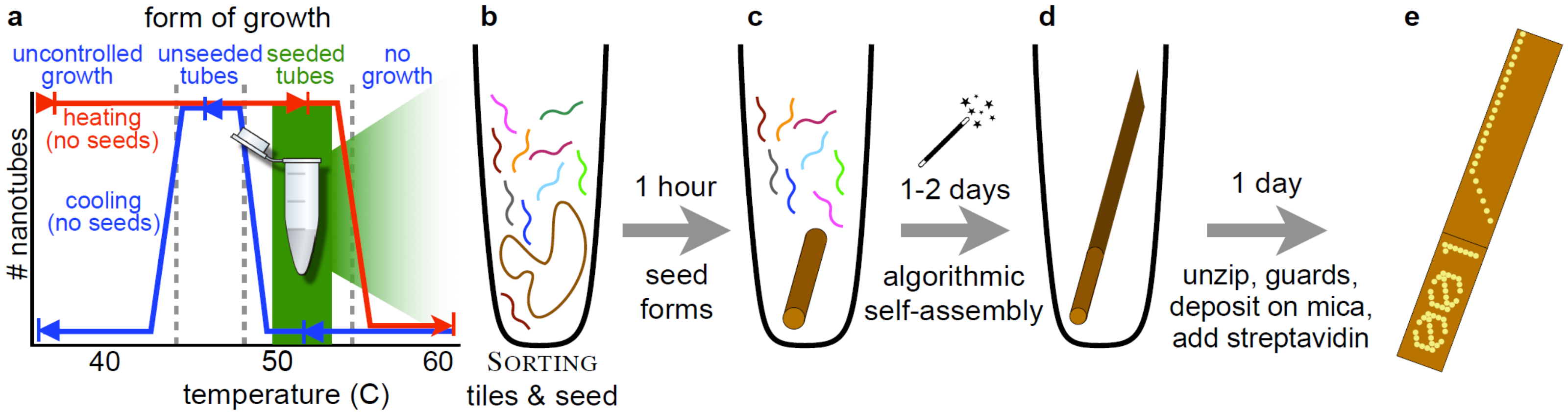
Generate
plate map

```
*****************************************************
* pos3=u, coding staples, no biotin (14 staples)
16H barrel stap 2                     |
  1  2  3  4  5  6  7  8  9 10 11 12 |
A                                     |
B                          X  X       |
C                    X                |
D                       X     X       |
E                          X  X       |
F                 X  X     X     X    |
G                       X  X  X       |
H                                     |
pos3=u, coding staples, biotin (16 staples)
16H 3bit biotin staples               | unzippers2 + biotin_staples_2_3      |
  1  2  3  4  5  6  7  8  9 10 11 12 |   1  2  3  4  5  6  7  8  9 10 11 12 |
A                          X  X     X | A                                    |
B                          X     X  X | B                                    |
C                          X     X  X | C                                    |
D                                X  X | D                                    |
E                          X     X    | E                       X            |
F                                     | F                       X            |
G                                     | G                                    |
H              X                      | H                                    |
*****************************************************
```

represents some combination of
circuit and input, e.g.,
013 = *“parity circuit, input=011010”*

label with
streptavidin

Experimental procedure



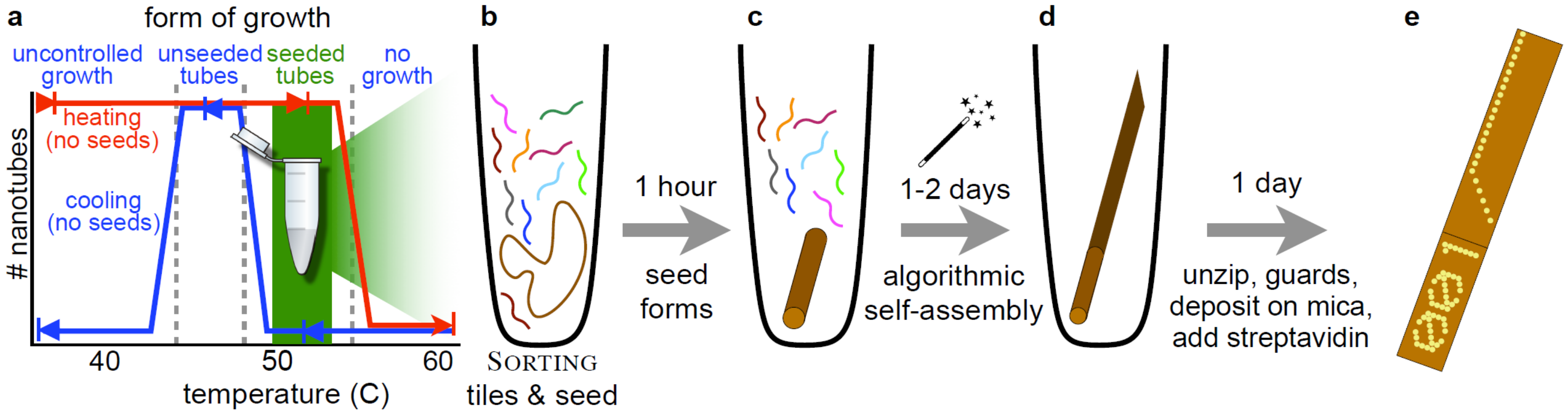
To execute circuit γ on input $x \in \{0,1\}^*$:

Mix

-)
- “adapter” strands encoding x
- tiles computing γ

- Anneal 90° C to 50.9° C in 1 hour (*origami seeds form*)
- Hold at 50.9° C for 1-2 days (*tiles grow tubes from seed*)
- Add “unzipper” strands (remove seam to convert tube to ribbon)
- Add “guard” strands (complements of output sticky ends, to deactivate tiles)
- Deposit on mica, buffer wash, add streptavidin, AFM

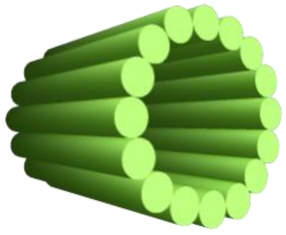
Experimental procedure



To execute circuit γ on input $x \in \{0,1\}^*$:

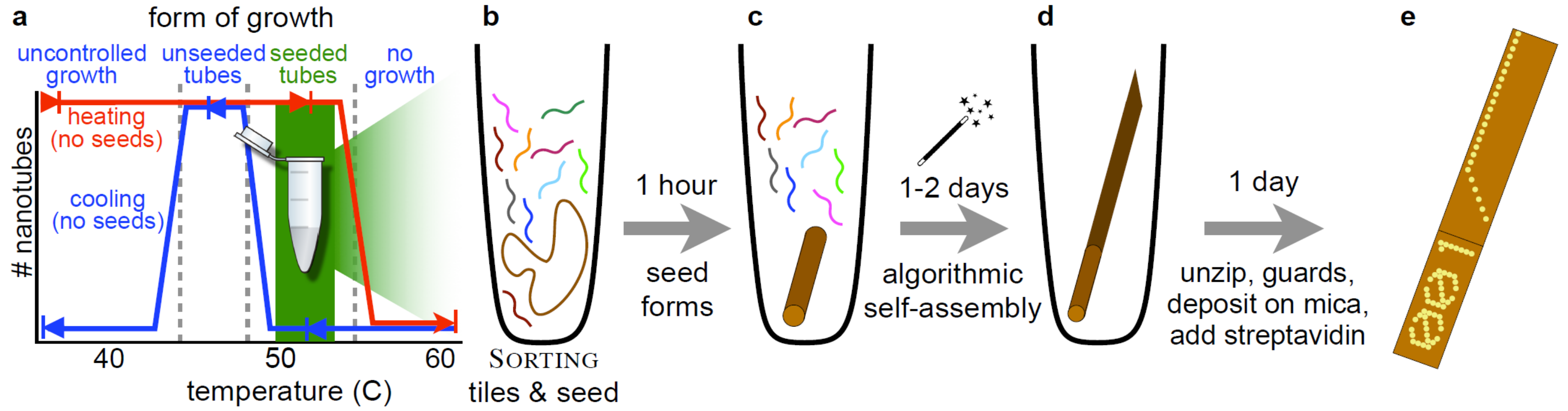
Mix

- origami (bar-coded to identify both γ and x)
- "adapter" strands encoding x
- tiles computing γ



- Anneal 90° C to 50.9° C in 1 hour (*origami seeds form*)
- Hold at 50.9° C for 1-2 days (*tiles grow tubes from seed*)
- Add "unzipper" strands (remove seam to convert tube to ribbon)
- Add "guard" strands (complements of output sticky ends, to deactivate tiles)
- Deposit on mica, buffer wash, add streptavidin, AFM

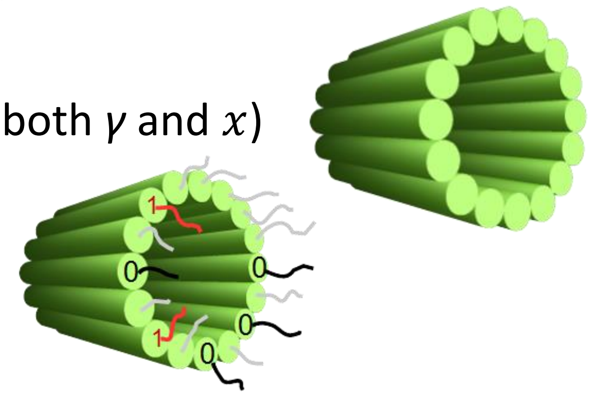
Experimental procedure



To execute circuit γ on input $x \in \{0,1\}^*$:

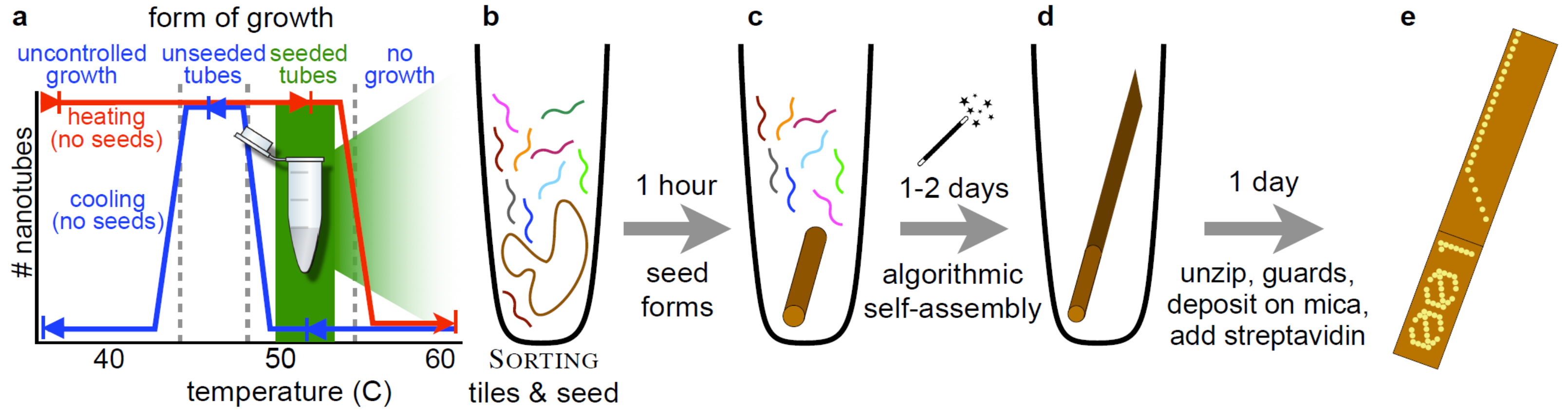
Mix

- origami (bar-coded to identify both γ and x)
- “adapter” strands encoding x
- tiles computing γ



- Anneal 90° C to 50.9° C in 1 hour (*origami seeds form*)
- Hold at 50.9° C for 1-2 days (*tiles grow tubes from seed*)
- Add “unzipper” strands (remove seam to convert tube to ribbon)
- Add “guard” strands (complements of output sticky ends, to deactivate tiles)
- Deposit on mica, buffer wash, add streptavidin, AFM

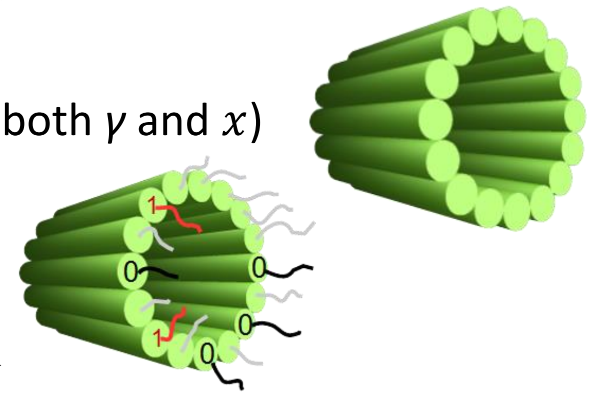
Experimental procedure



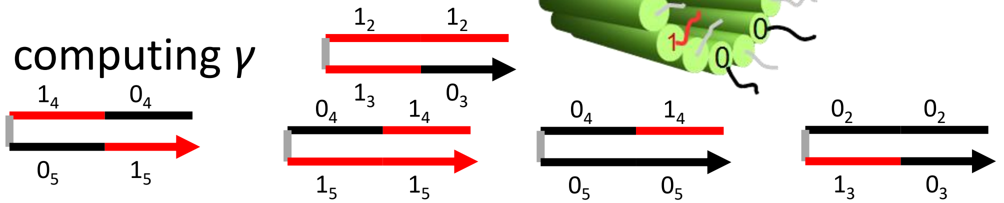
To execute circuit γ on input $x \in \{0,1\}^*$:

Mix

- origami (bar-coded to identify both γ and x)
- "adapter" strands encoding x

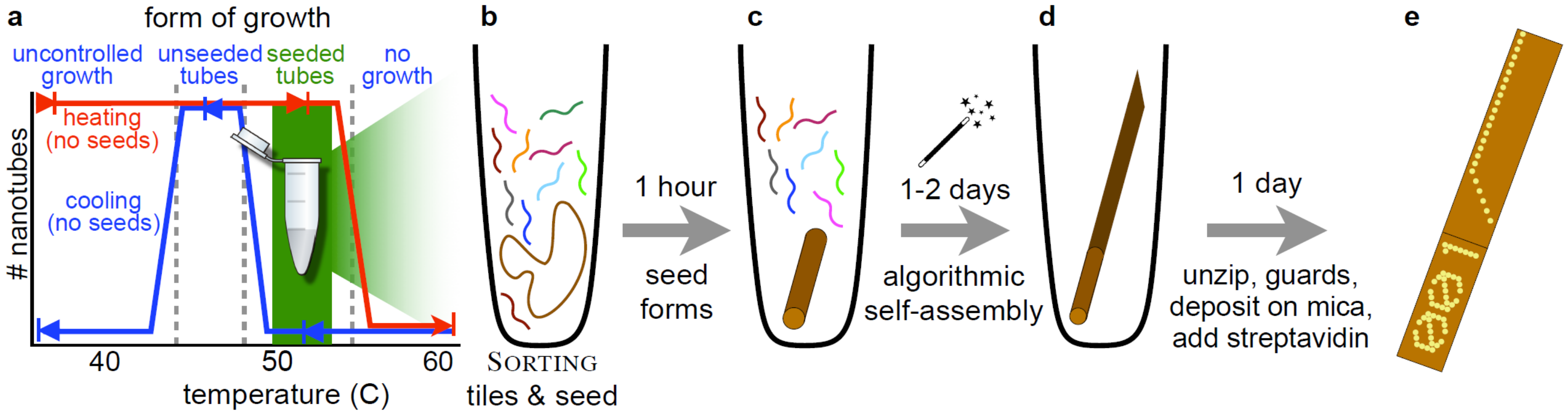


- tiles computing γ



- Anneal 90° C to 50.9° C in 1 hour (*origami seeds form*)
- Hold at 50.9° C for 1-2 days (*tiles grow tubes from seed*)
- Add "unzipper" strands (remove seam to convert tube to ribbon)
- Add "guard" strands (complements of output sticky ends, to deactivate tiles)
- Deposit on mica, buffer wash, add streptavidin, AFM

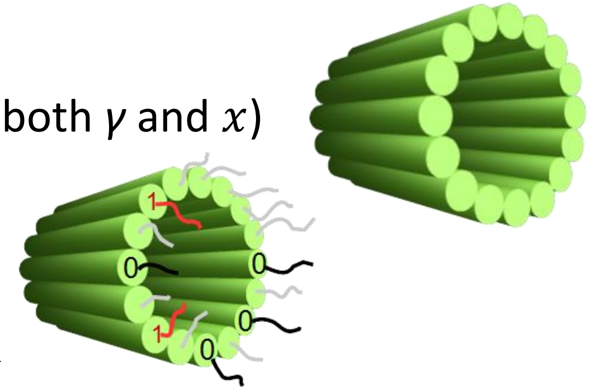
Experimental procedure

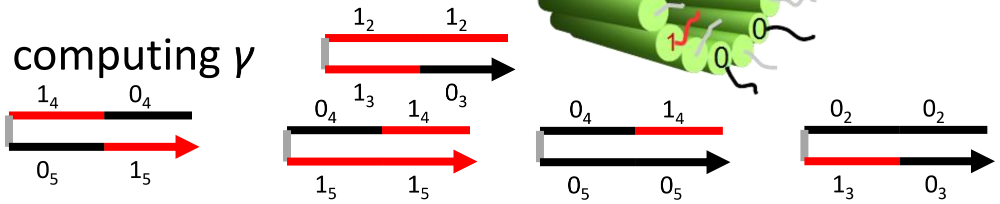


To execute circuit γ on input $x \in \{0,1\}^*$:

Mix

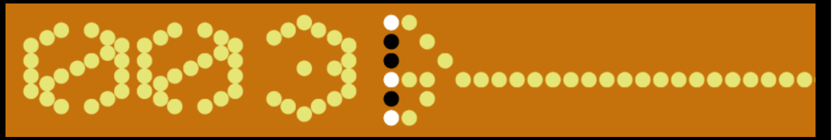
- origami (bar-coded to identify both γ and x)
- "adapter" strands encoding x
- tiles computing γ





- Anneal 90° C to 50.9° C in 1 hour (*origami seeds form*)
- Hold at 50.9° C for 1-2 days (*tiles grow tubes from seed*)
- Add "unzipper" strands (remove seam to convert tube to ribbon)
- Add "guard" strands (complements of output sticky ends, to deactivate tiles)
- Deposit on mica, buffer wash, add streptavidin, AFM

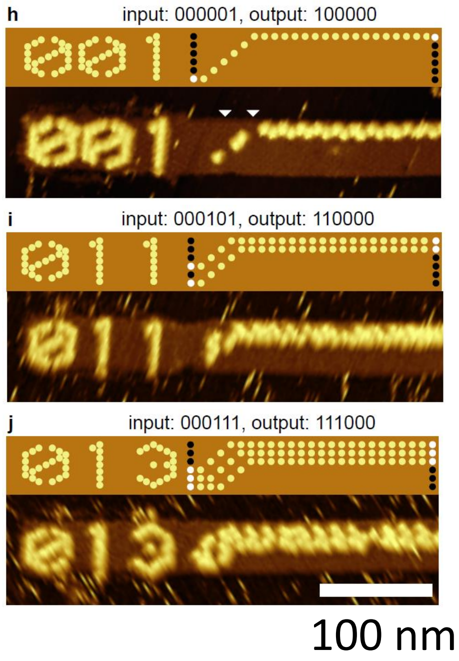
```
def test_parity_100101():
    actual = parity('100101')

    expected =
```



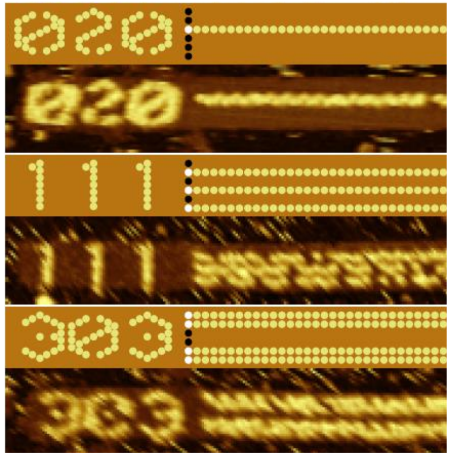
```
    assertEquals(actual, expected)
```

Results

SORTING

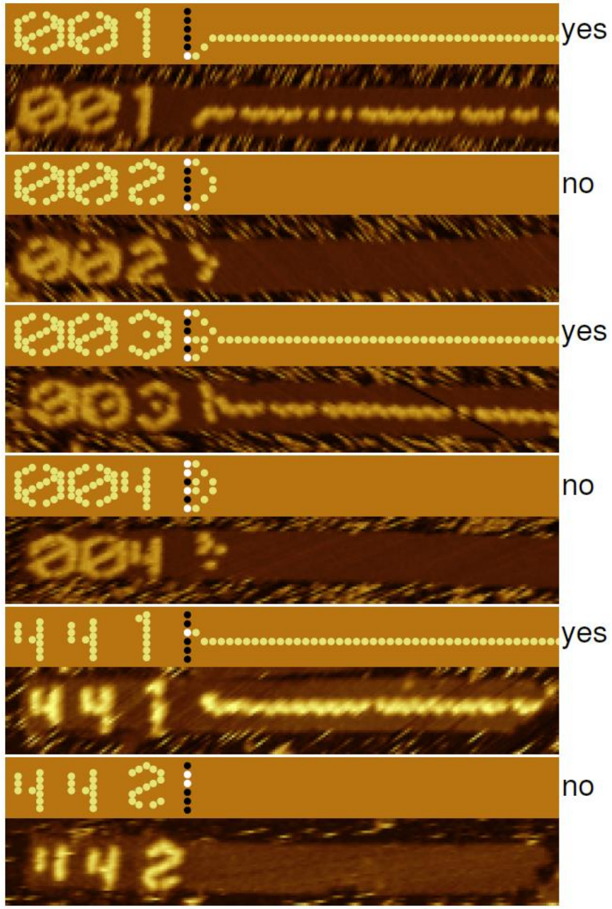


COPY



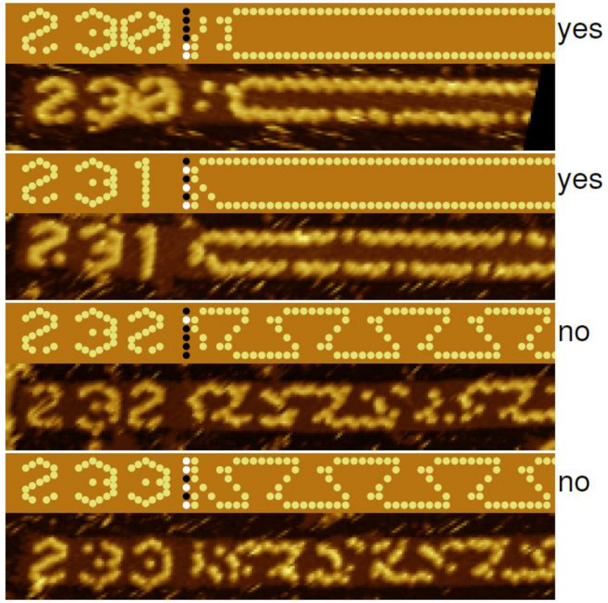
PARITY

Is the number of 1's odd?



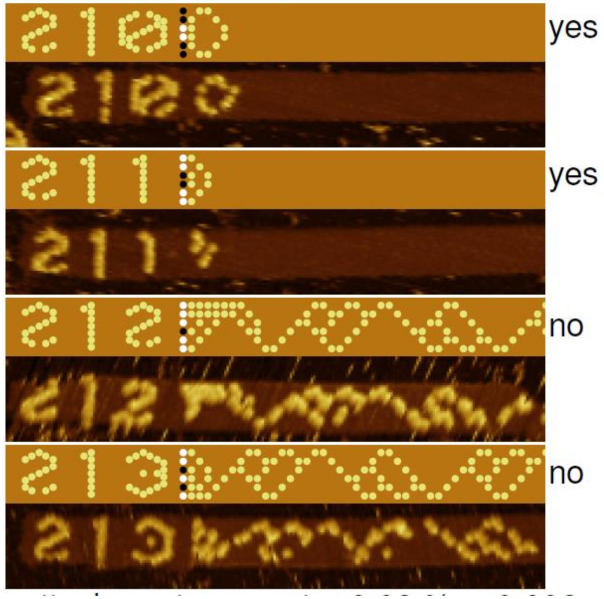
MULTIPLEOF3

Is the input binary number a multiple of 3?



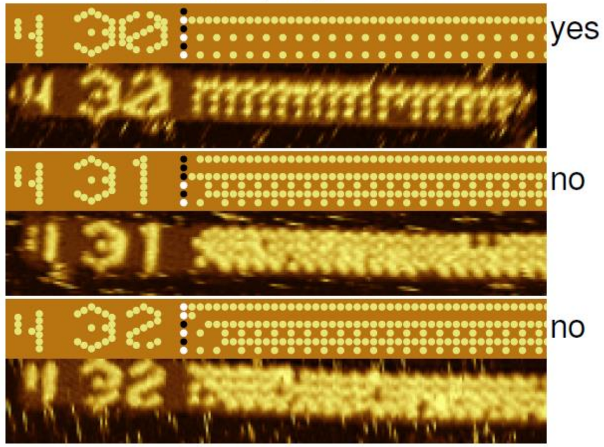
PALINDROME

Is the input a palindrome?



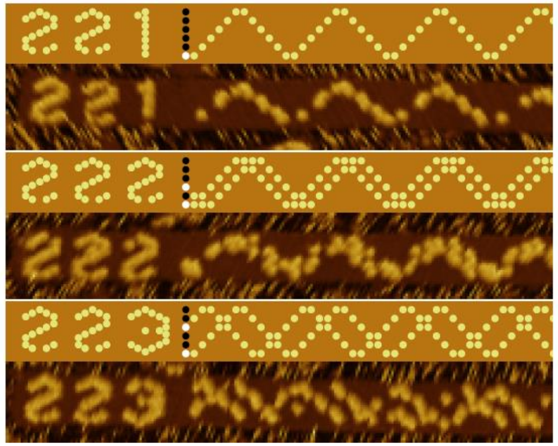
RECOGNISE21

Is the binary input = 21?

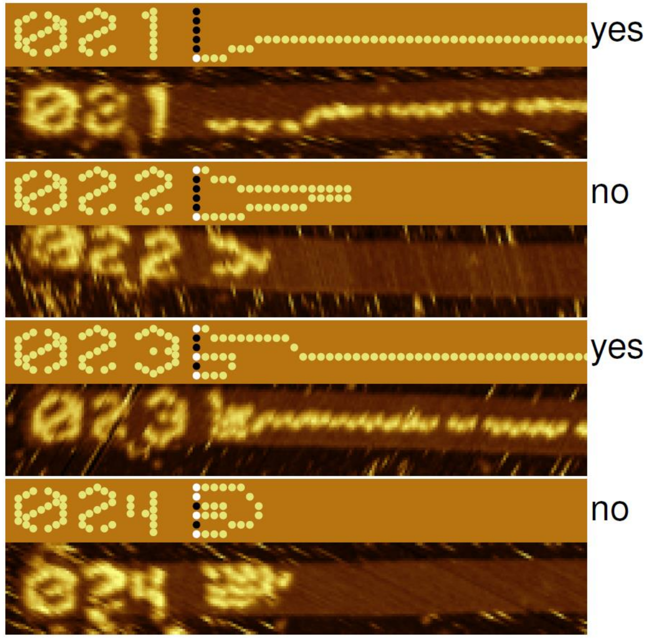


ZIG-ZAG

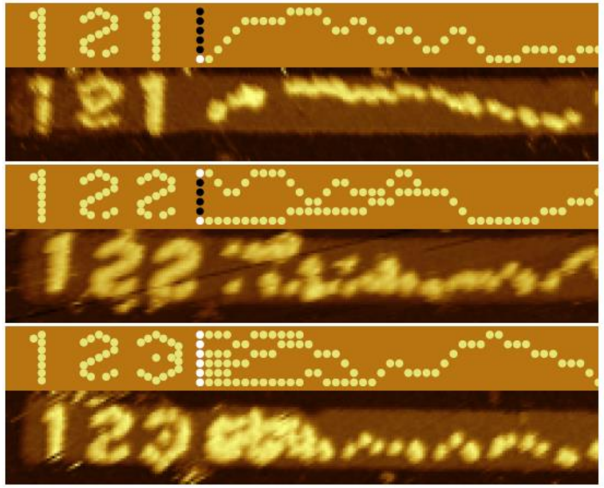
Repeating pattern



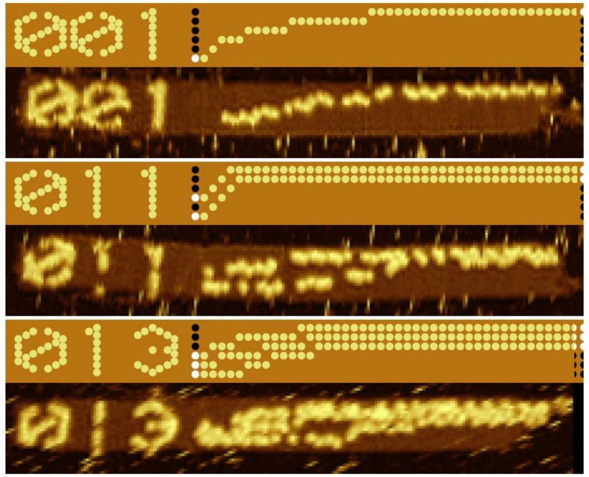
LazyParity



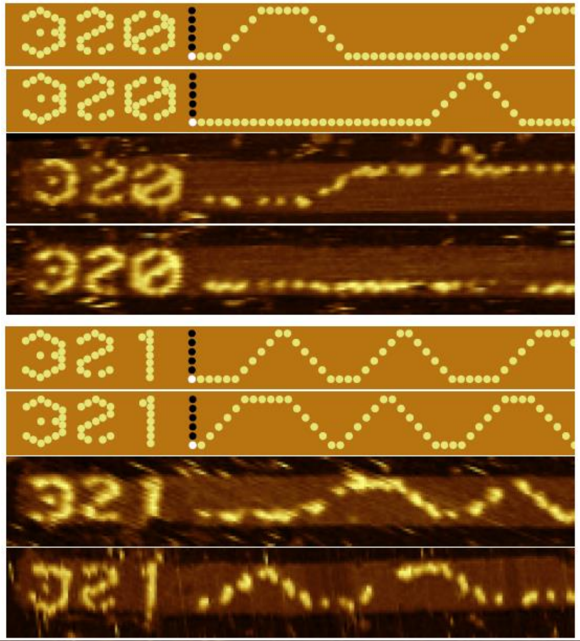
LeaderElection



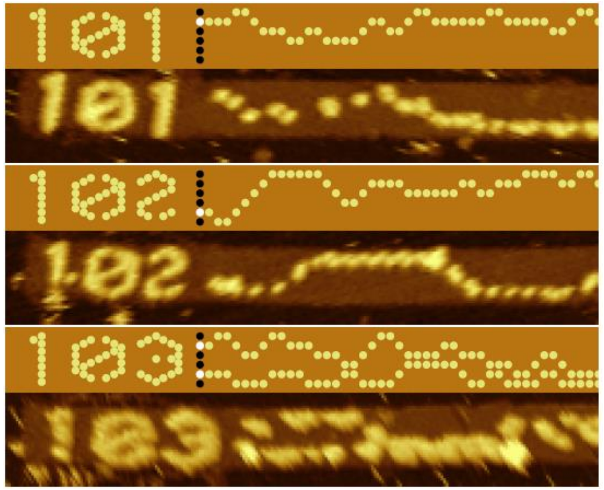
LazySorting



Waves

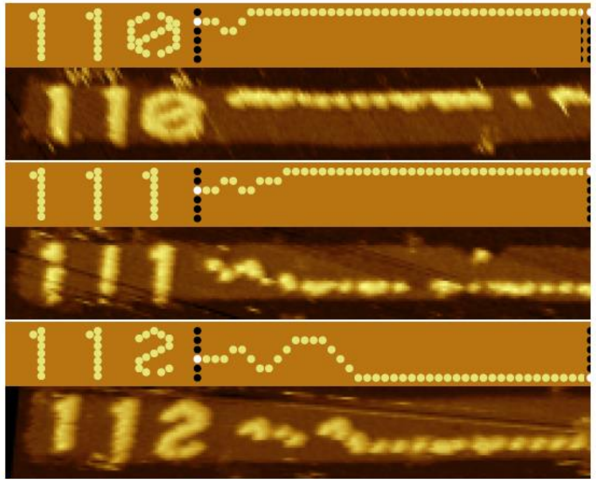


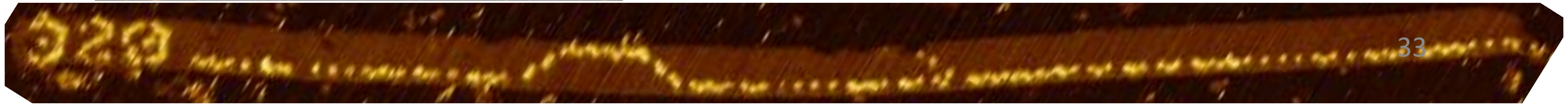
RandomWalkingBit



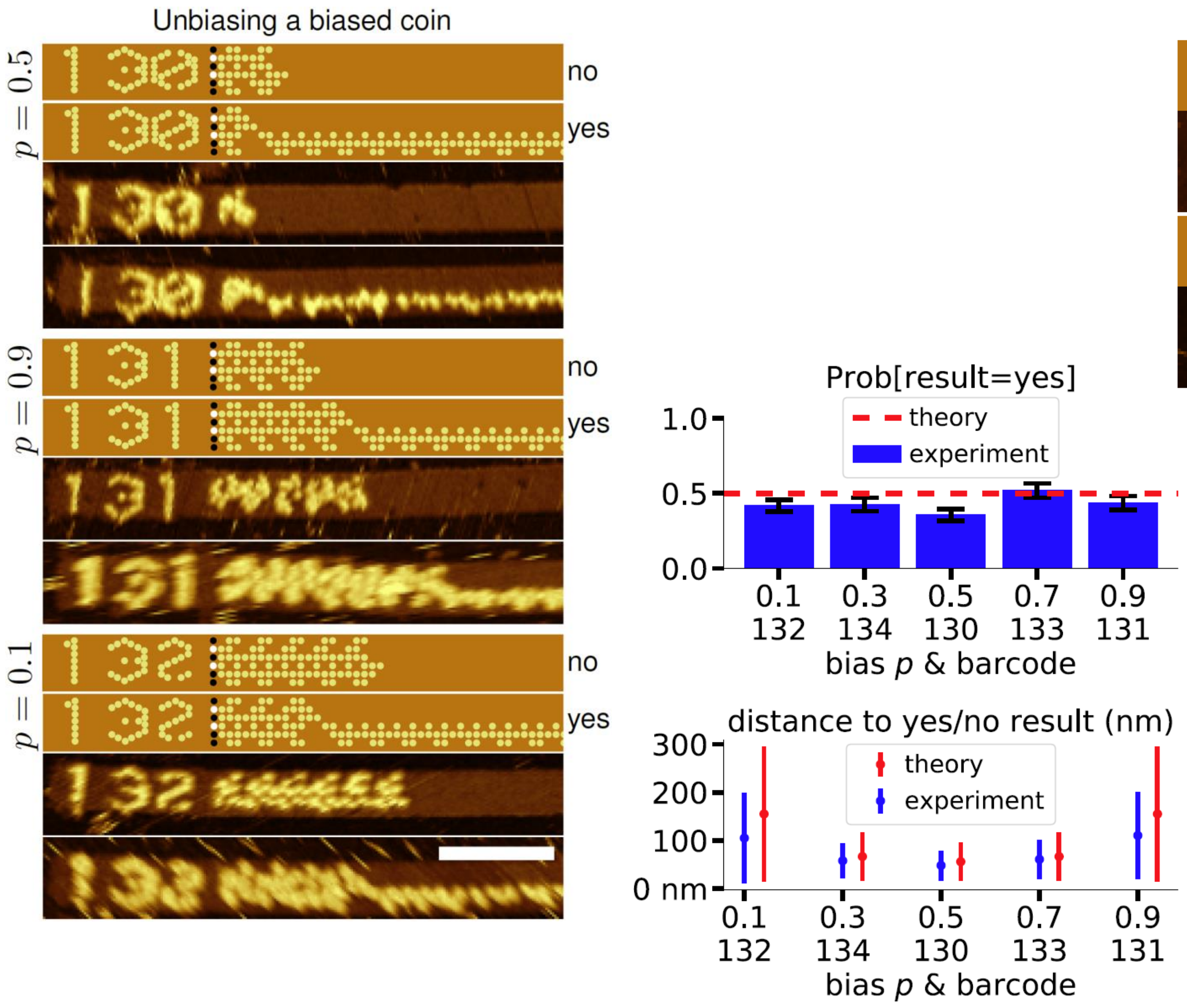
AbsorbingRandomWalkingBit

Random walker absorbs to top/bottom

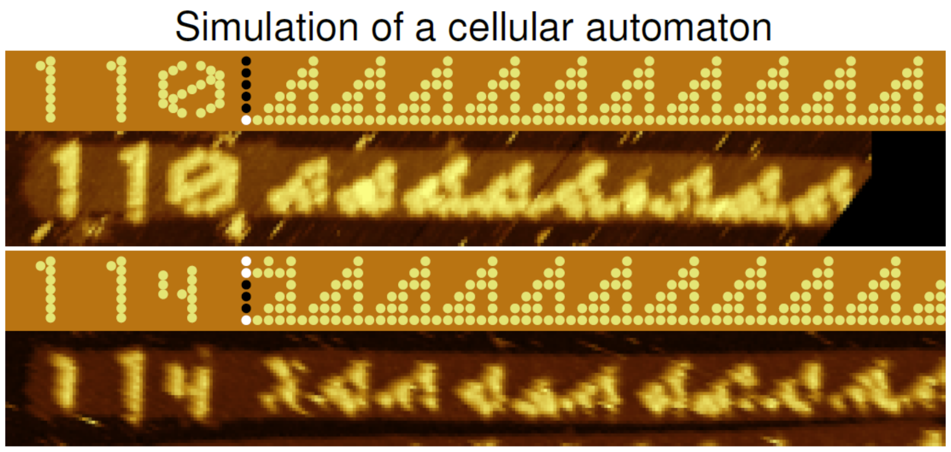




FAIRCOIN



RULE110



Counting to 63

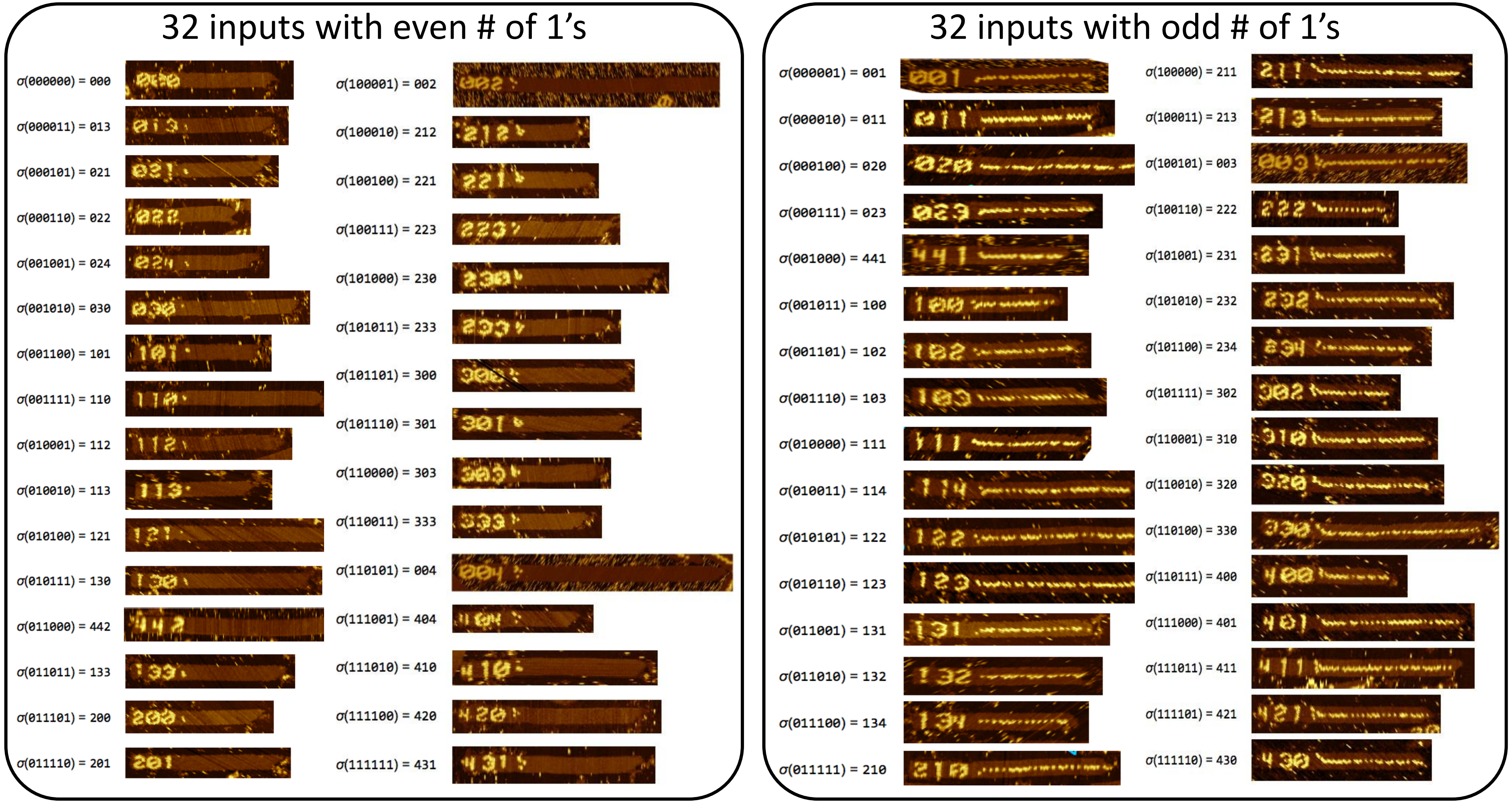
Circuit with 63 distinct outputs

Is there a 64-counter?

No!
Proof by Tristan Stérin, ENS Lyon & Inria
(Consequence of following theorem: *Any bijective Boolean circuit having one output bit that does not depend on all of its input bits cannot compute odd bijections.*)

Parity tested on all inputs

2^6 = 64 inputs with 6 bits

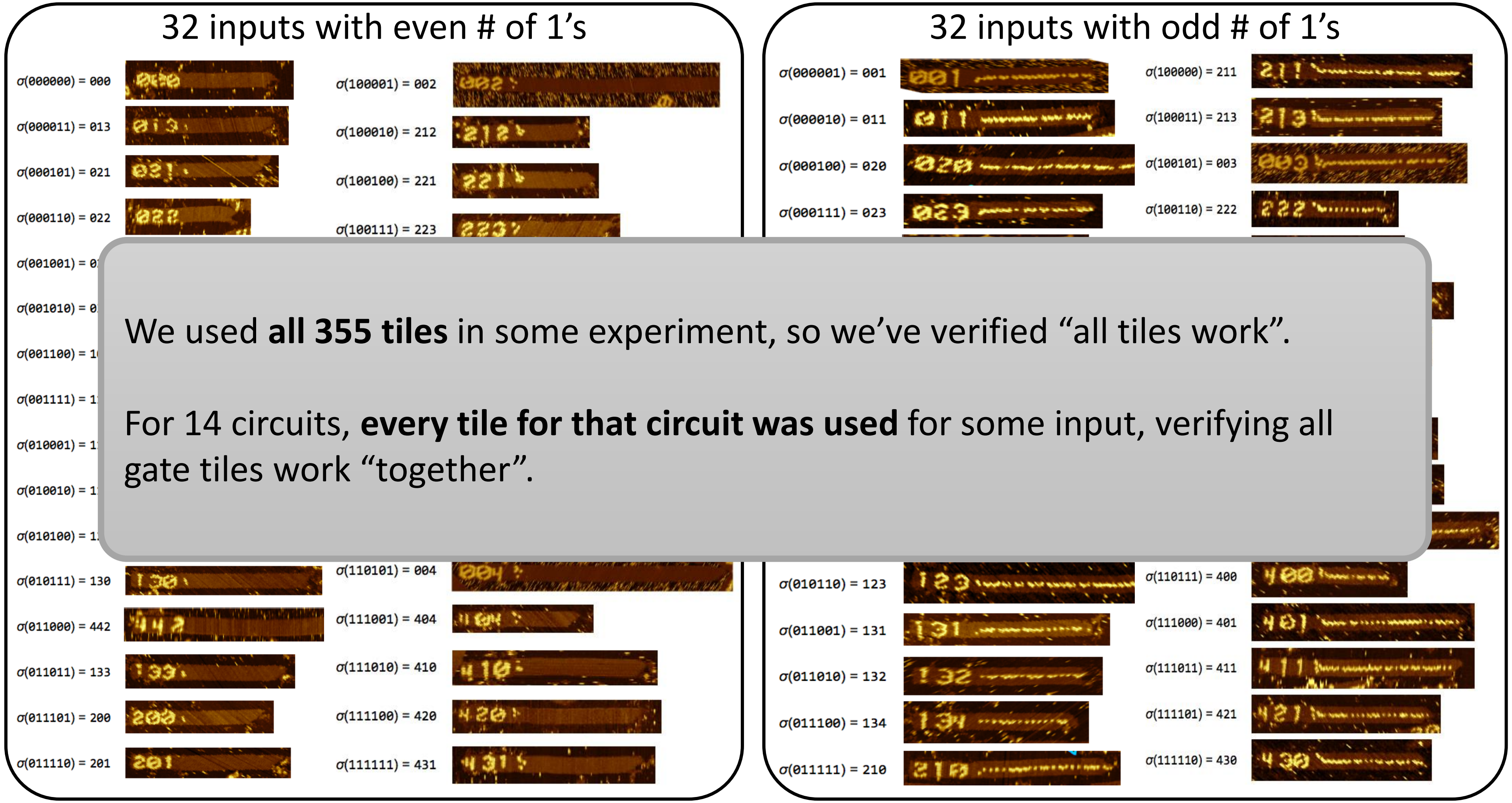


σ(6-bit input) = 3-digit barcode representing that input

150 nm

Parity tested on all inputs

2^6 = 64 inputs with 6 bits



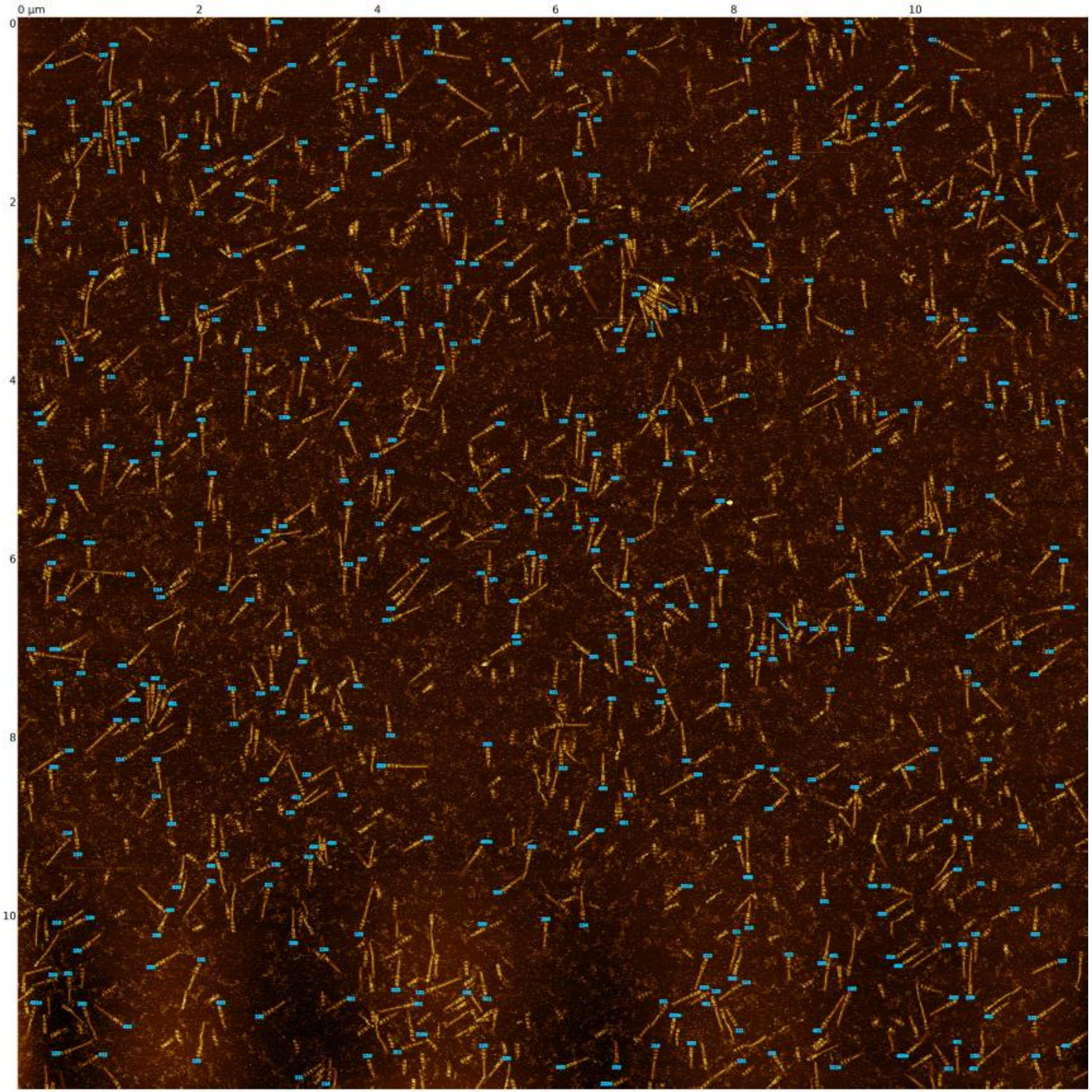
We used **all 355 tiles** in some experiment, so we've verified "all tiles work".

For 14 circuits, **every tile for that circuit was used** for some input, verifying all gate tiles work "together".

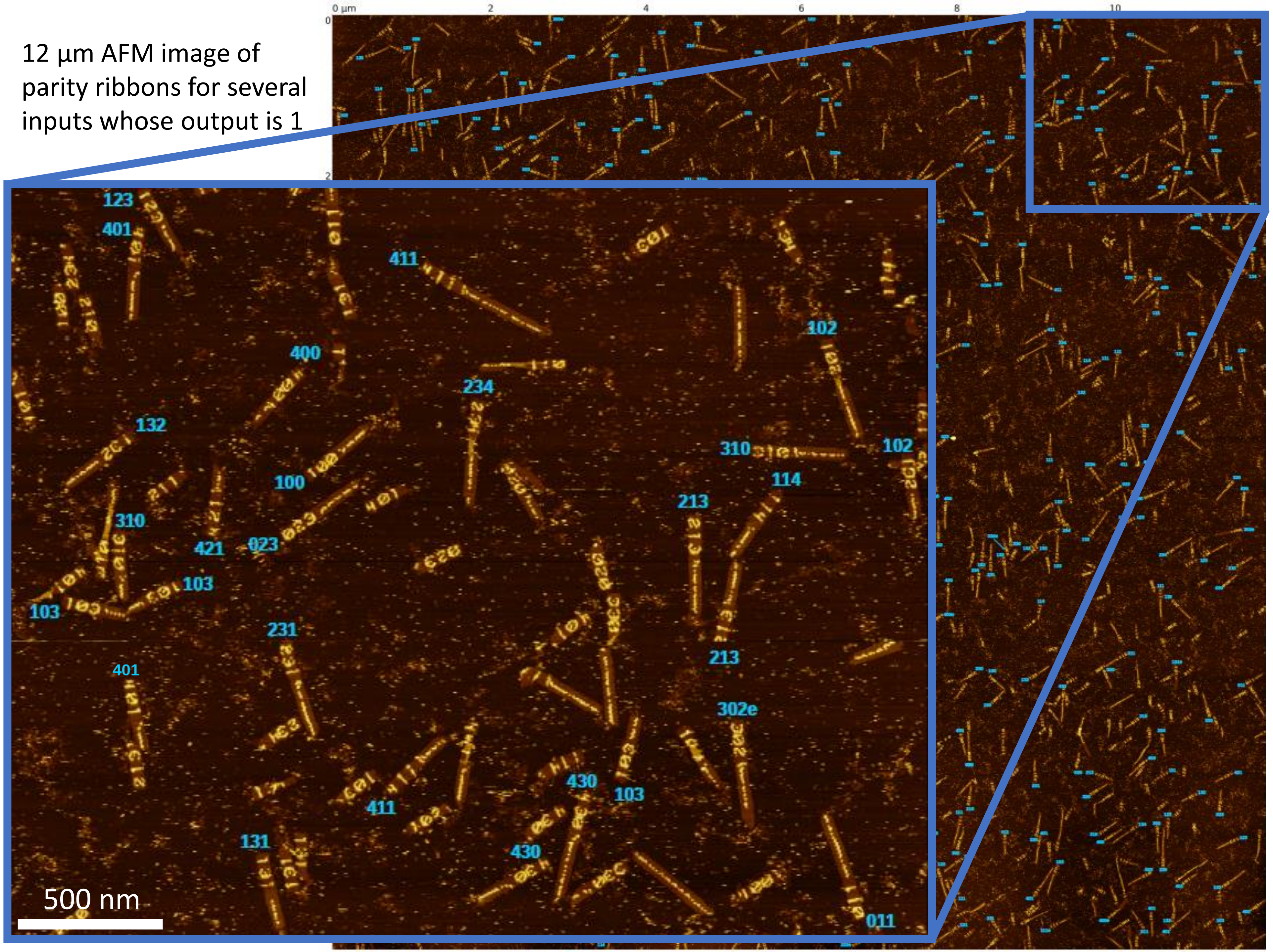
σ(6-bit input) = 3-digit barcode representing that input

150 nm

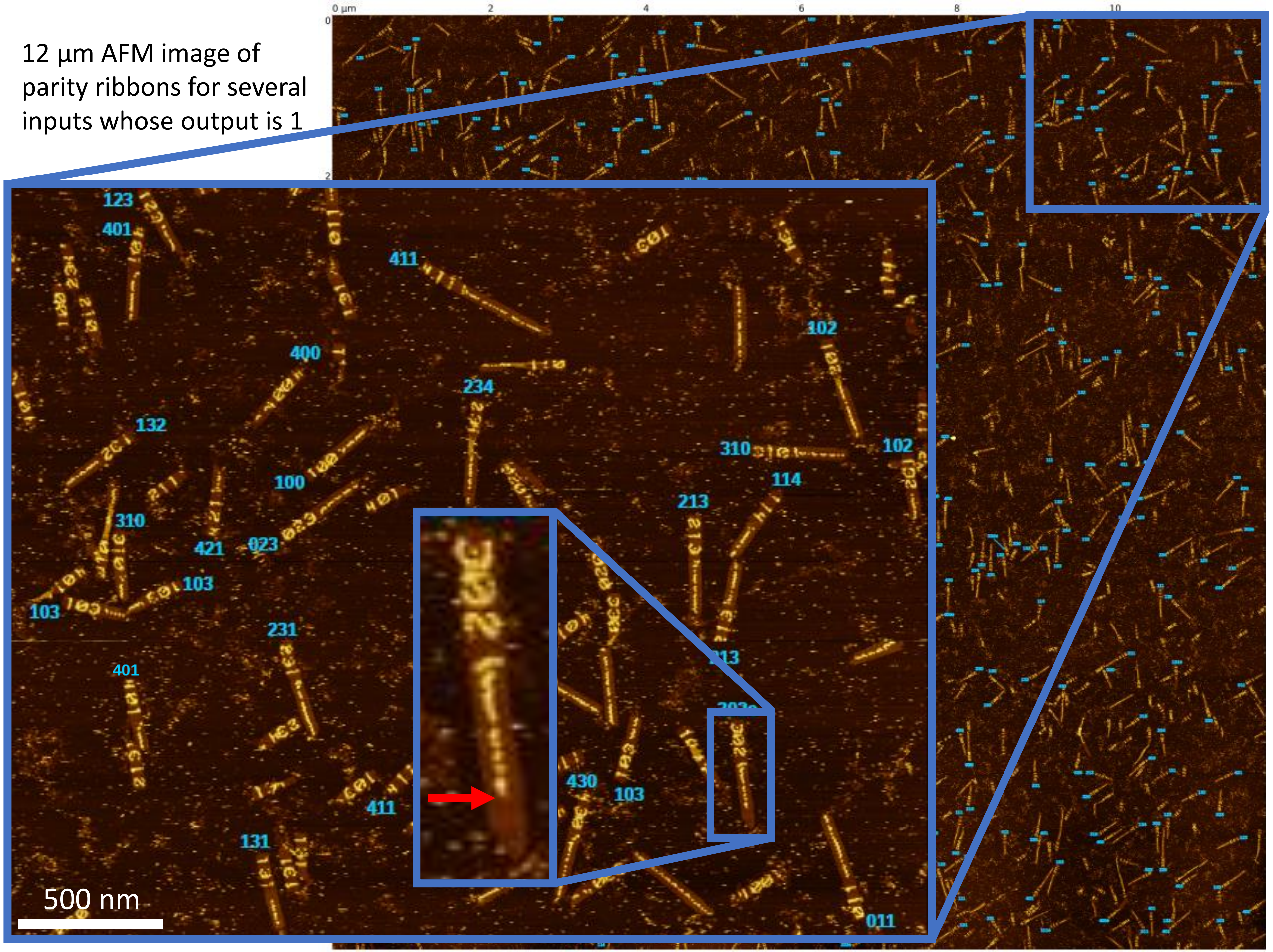
12 µm AFM image of parity ribbons for several inputs whose output is 1



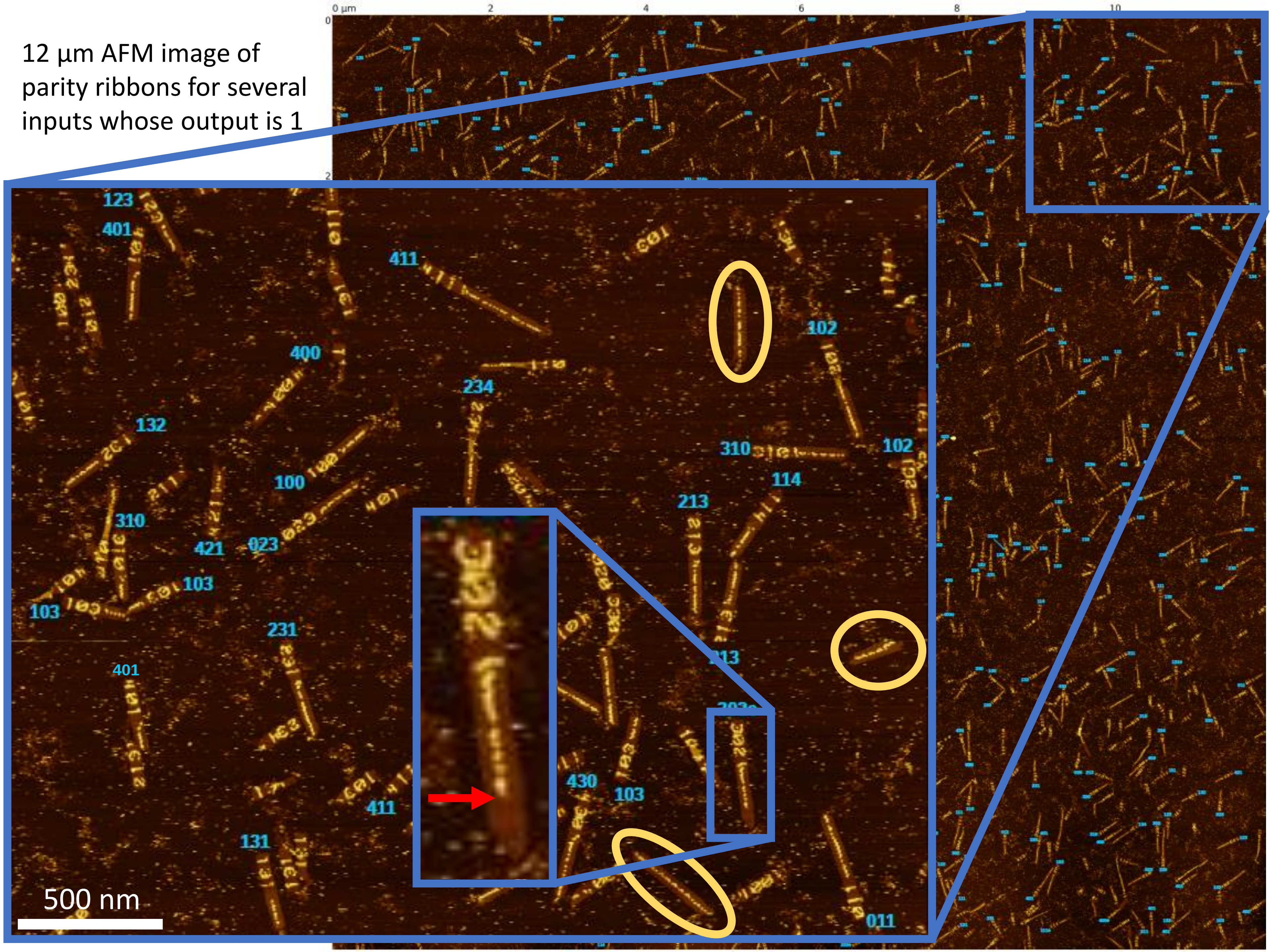
12 µm AFM image of parity ribbons for several inputs whose output is 1



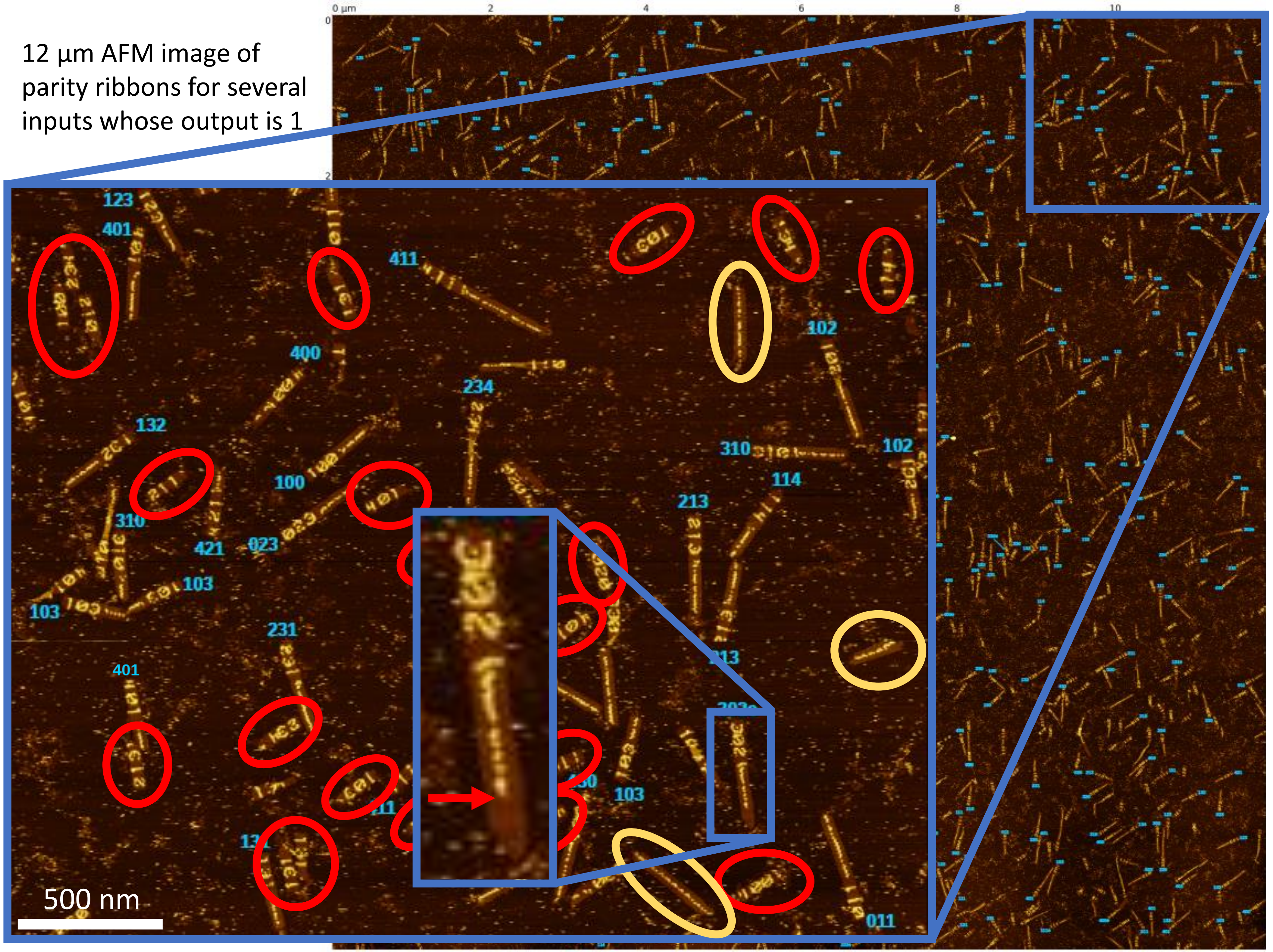
12 µm AFM image of parity ribbons for several inputs whose output is 1



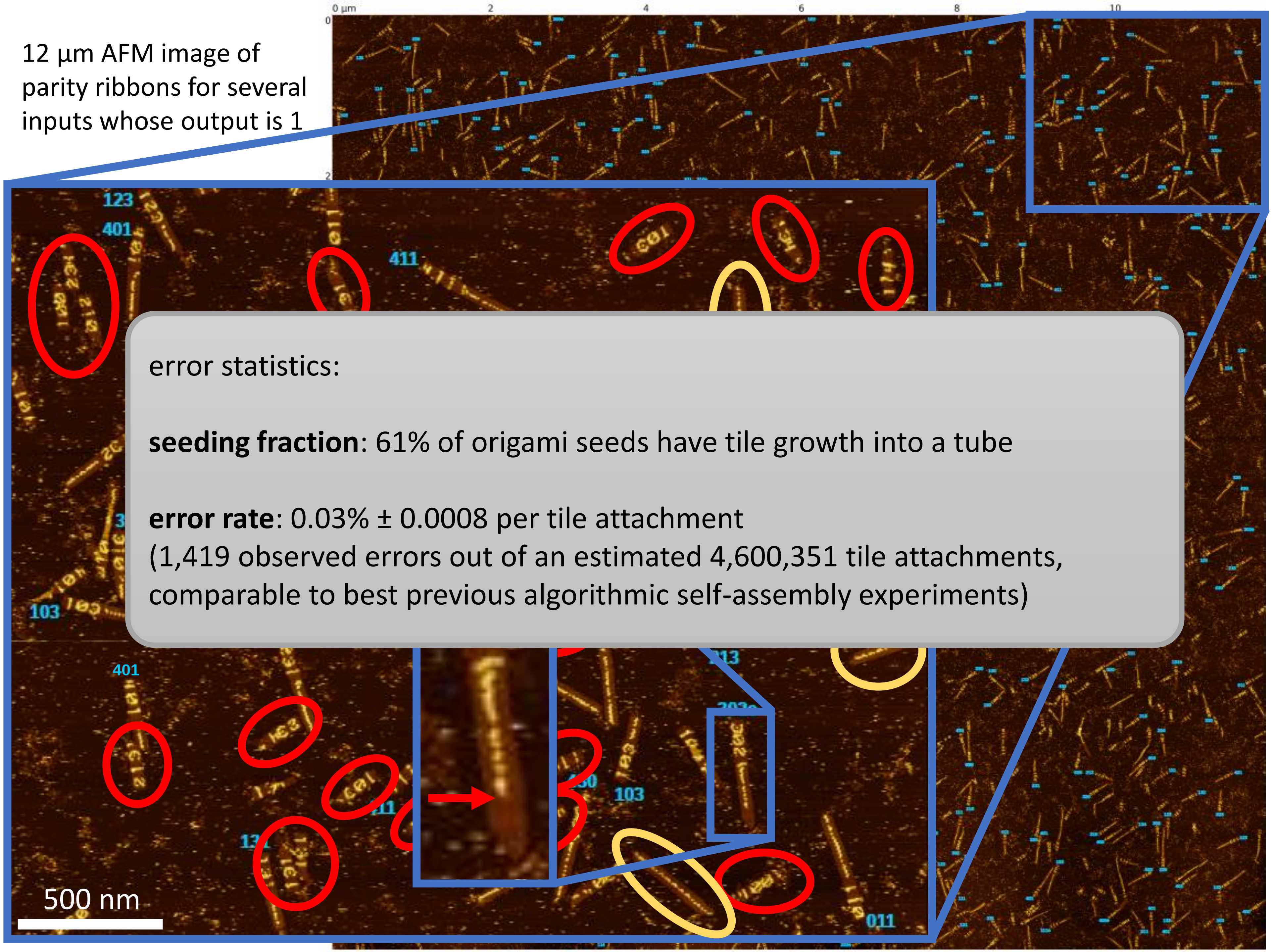
12 µm AFM image of parity ribbons for several inputs whose output is 1



12 µm AFM image of parity ribbons for several inputs whose output is 1



12 µm AFM image of parity ribbons for several inputs whose output is 1



error statistics:

seeding fraction: 61% of origami seeds have tile growth into a tube

error rate: 0.03% ± 0.0008 per tile attachment
(1,419 observed errors out of an estimated 4,600,351 tile attachments, comparable to best previous algorithmic self-assembly experiments)

Next big challenge: Algorithmically control shape

We “drew” interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Next big challenge: Algorithmically control shape

We "drew" interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.

[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Next big challenge: Algorithmically control shape

We "drew" interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.

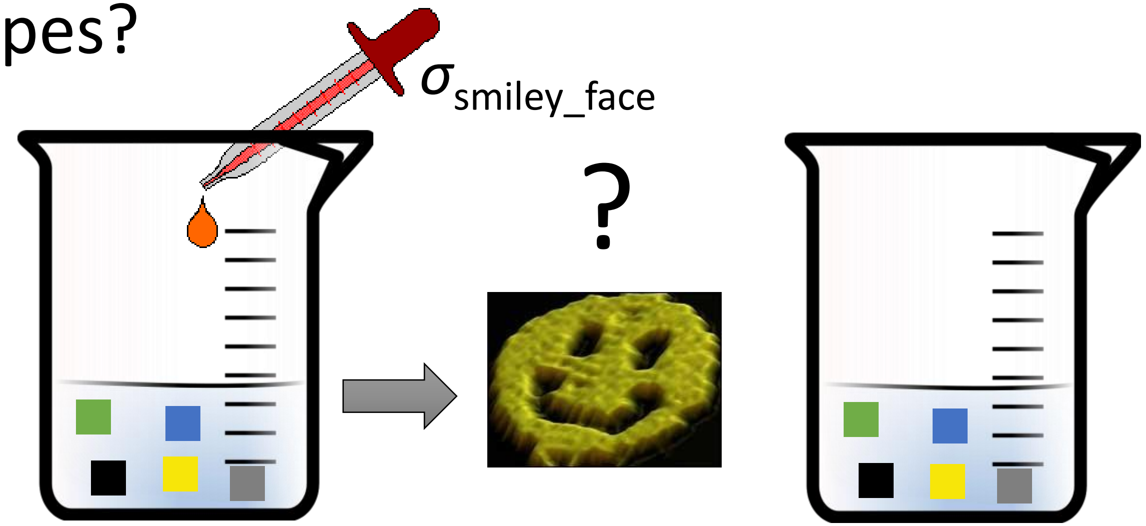
[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Next big challenge: Algorithmically control shape

We "drew" interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.



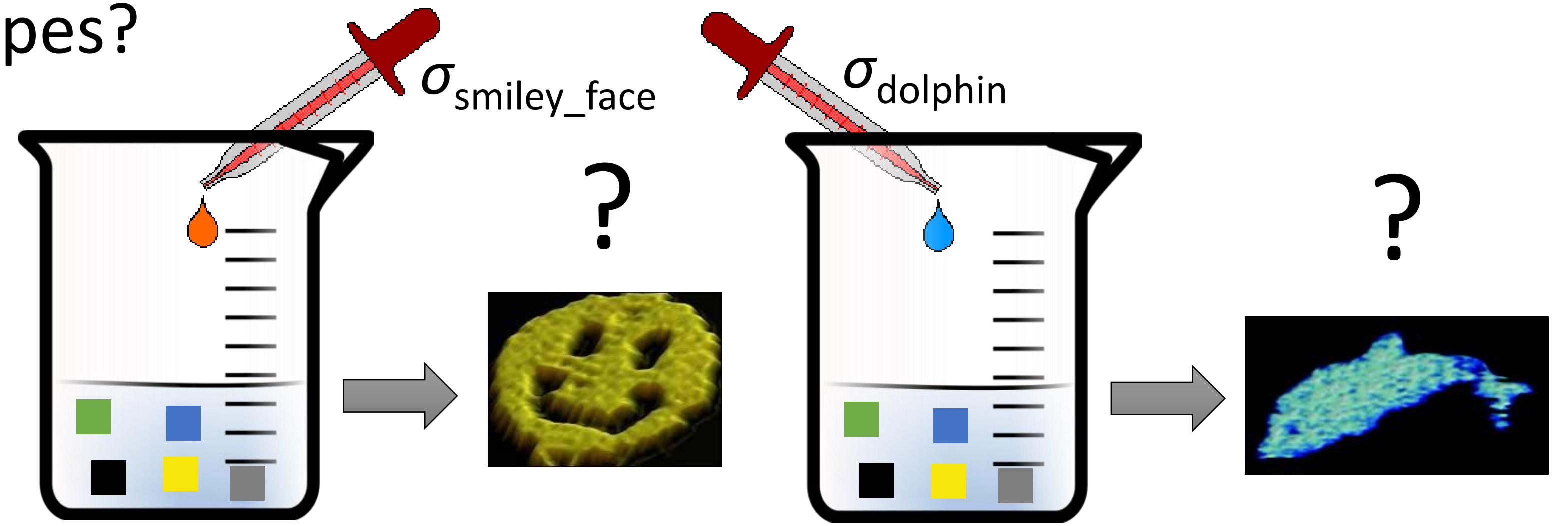
[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Next big challenge: Algorithmically control shape

We "drew" interesting patterns on a boring shape (infinite rectangle)

Can we grow interesting shapes?

Theorem: There is a single set T of tile types, so that, for any finite shape S, from an appropriately chosen seed σ_S "encoding" S, T self-assembles S.



These tiles are universally programmable for building any shape.

[*Complexity of Self-Assembled Shapes*. Soloveichik and Winfree, SIAM Journal on Computing 2007]

Thank you!